A REACTIVE APPROACH FOR USE-BASED PRIVACY

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Eleanor Jane Birrell

August 2018

A REACTIVE APPROACH

FOR

USE-BASED PRIVACY

Eleanor Jane Birrell, Ph.D.

Cornell University 2018

This dissertation describes an investigation into the feasibility of expressing and enforcing use-based privacy, which posits that privacy can be provided by preventing harmful uses of sensitive information. Use-based privacy is shown to benefit from a *reactive* policy language—one that specifies not only a current set of restrictions but also describes how those restrictions change. An instantiation of such a reactive language, the Avenance language, is defined; its expressiveness is demonstrated with real-world policies. The dissertation also explores the feasibility of a technical means for enforcing compliance. Systems are described for facilitating policy compliance by benign principals and architectures for enforcing policy compliance in the presence of adversarial principals. A policy provider that associates use-based privacy policies with sensitive values as *policy tags* also is described. The described work collectively constitutes strong evidence for the feasibility of use-based privacy.

BIOGRAPHICAL SKETCH

Eleanor Birrell was born and raised in California. She attended Harvard University, where she discovered that computer science presents lots of fun puzzles; she graduated with a B.A., *cum laude*, in Computer Science and Mathematics in 2009.

After several years spent playing with questions related to cryptography, security, and privacy, she received an M.S. in Computer Science from Cornell University in 2014. She also completed a minor in Law. And, after several more years, she finally wrote a dissertation.

To my family,

who always believed this dissertation would eventually exist

ACKNOWLEDGEMENTS

Many thanks to my advisor, Fred Schneider, for hours of insightful discussions, careful feedback, and infinite patience. This dissertation, and the work it describes, would never exist without you.

I would also like to thank my other collaborators: Robbert van Rennesse, Anders Gjerdrum, Magnus Stenhaug, Håvard Johansen, and Dag Johansen. Much of the work described in this dissertation was joint work that benefited from your contributions. Notably, Håvard was the driving force behind the LoNet project, Anders was an invaluable collaborator on the SGX work, and Robbert was always happy to provide advice on experimental design and the more systems-y components. The minor members of my committee—Andrew Myers, Rafael Pass, and Mitchel Lasser—also contributed insightful questions and helpful feedback on this dissertation.

I am eternally grateful to my family and friends for all their support through the years. You were always there for me, providing technical insights, moral support, and much-needed distractions.

This work was supported in part by a National Science Foundation Graduate Research Fellowship (2010-2013), by AFOSR grant F9550-16-0250, and by NSF grant 1642120.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Current approaches to privacy in networked information systems are poorly suited to modern networked information systems, where information is collected without user awareness, and data sharing and analysis are pervasive. This dissertation explores the feasibility of an alternate view, sometimes called *use-based privacy* [14, 15, 46], which equates privacy with preventing harmful uses.

Instead of requiring informed consent from data subjects, use-based privacy assumes there has been a societal evaluation that has identified harmful uses. This evaluation presumably will have balanced potential harms and potential benefits of information use—and evaluated the countermeasures in place to prevent potential harms—to determine which uses should be deemed harmful. A system that avoids harmful uses is then considered *privacy-compliant*.

Use-based privacy differs from most previous views of privacy in three key ways:

- Use-based privacy policies do not depend on the individual preferences of the data subject but instead focus on the collective.
- Use-based privacy policies describe how information may be used rather than limiting access or transmission.
- Use-based privacy policies impose restrictions on how both raw data and derived data may be used, and thereby govern information flow through a system.

The first aspect of use-based privacy is similar to the philosophy of contextual integrity [47, 48, 2], which defines privacy for personal information relative

to an appropriate context. Whether a context is appropriate is presumed to be determined by socially-defined informational norms, which might depend on time, location, purpose, and/or participating principals. So contextual integrity, like use-based privacy, moves away from user-defined policies and informed consent, focusing instead on elimination of harmful uses (as defined by informational norms). However, contextual integrity ignores how sensitive information is used as it flows through a networked information system and thus ignores derived data; it instead focuses on mediating individual communications and evaluating whether each data transmission is authorized.

In this dissertation, we investigate the feasibility of expressing and enforcing privacy regulations and corporate privacy policies; in doing so, we assume that such use restrictions are the result of the assumed societal evaluation balancing harms versus benefits. As the first step of our investigation, we analyze examples drawn from data-use contracts, existing privacy policies, and U.S. regulations to identify key attributes of a language for specifying use-based privacy policies in networked information systems.

We observe that *reactive labels* [37] are a natural way to realize the identified attributes. A reactive label maps a sequence of operations (which describe the derivation of the value) to a set of restrictions on how that resulting value may be used. When use-based privacy policies are specified as reactive labels, policy specifications can be attached to data values and these specifications will flow to derived values as information flows through the system. Existing privacy policy specification languages are not reactive, so we introduce a new privacy specification language—the *Avenance language*—that uses reactive labels and is well-suited for expressing use-based privacy. The argument for a reactive approach to use-based

privacy, the Avenance language, and our experience using the Avenance language to describe real-world policies are described in Chapter 2.

As the next step, we explore how policy compliance might be enforced. Use-based privacy expresses restrictions on how sensitive values may be used; adversaries are principals that use (potentially) sensitive values—termed *service providers*. We consider three different threat models: *benign service providers*, *accountable service providers*, and *malicious service providers*.

Benign service providers are presumed to have non-technical incentives to comply with all relevant policies (e.g., concern for reputation or legal consequences).[1] Such service providers might violate a policy due to a mistake or a programmer error, but they are not malicious. So the goal of an enforcement mechanism here is to facilitate policy compliance by well-intentioned, but imperfect, actors. We describe two different systems for enforcing use-based privacy policies in this threat model: LoNet and Tir. LoNet is a runtime system that augments files with Avenance policies and enforces file-granularity policy compliance by modifying the file system. Tir is a middleware layer that provides an inline monitoring API for automatically checking Avenance policy compliance at the granularity of individual values. These tools are described in Chapter 3.

To ensure policy compliance in the presence of accountable or malicious service providers, use-based privacy policies must be enforced whenever a service provider uses a value. This can be accomplished by a monitor that implements complete mediation, provided that monitor enforces the relevant policies. And to guarantee that the monitor is trustworthy, we need some means for monitoring behavior by

[1]Since existing real-world policies, including legal regulations (e.g., HIPAA) and data use policies (e.g., Facebook's site privacy policy) are legally-binding, assuming benign service providers is likely to be a reasonable threat model in many cases.

service providers.

Recent developments in trusted hardware—e.g., Intel's Software Guard Extensions (SGX) [19]—offer a new basis for placing trust in a monitor or other program. Using SGX, an untrusted principal can provide a remotely-authenticatable proof or *quote* which attests that some program is running or has produced a given output. Chapter 4 explores how to leverage that root to enforce use-based privacy in the presence of accountable or malicious service providers.

Our investigation into the feasibility of expressing and enforcing use-based privacy has thus far assumed that policy specifications are attached to values as *tagged values* that flow through a networked information system. As a final step in this investigation, we explore how to initially associate policy specifications with values. Drawing on lessons from the literature on identity management systems, Chapter 5 describes an approach that relies on a system of decentralized policy providers to associate policies with values. We also describe the implementation of one possible policy provider.

Widespread adoption of use-based privacy will likely require further investigation into the tradeoffs of various mechanisms for expressing and enforcing use-based privacy in real systems. Nonetheless, the described work collectively constitutes strong evidence for the feasibility of use-based privacy, and we view this work a promising step towards developing a privacy-enhancing, use-based privacy ecosystem for the modern world.

CHAPTER 2

EXPRESSING USE-BASED PRIVACY

We begin by discussing in Section 2.1 why observing that a reactive language is well-suited for use-based privacy. This observation is supported by two independent pieces of evidence: Drawing on examples from data-use contracts, existing privacy policies, and U.S. regulations, we identify four attributes inherent to use-based privacy—a use-based privacy specification language must be data-centric, it must be provenance-dependent[1] it must admit both permissions and obligations, and it must include both sticky and local restrictions. All are fulfilled by a reactive language. Independently, a user study performed on Mechanical Turk shows that collective user preferences are in fact well-matched to a reactive policy specification language.

In Section 2.2, we suggest a reactive privacy specification language—the Avenance language—that is well-suited for expressing use-based privacy. Avenance policies give restrictions in terms of who is using the data, the type of use (i.e., which operation accesses the data), and the purpose of that use. The manner in which an Avenance policy's current authorizations change is encoded as a *privacy automaton*, a finite state automata inspired by RIF automata [37].

We evaluated the Avenance language by using it to specify real-world privacy policies. We encoded the full set of privacy requirements defined by a pair of representative example policies—the Health Information Portability and Accountability Act (HIPAA) [32] and Facebook's site privacy policy [25]—as Avenance policies.

[1]Provenance-dependent policies that always increase permissions are sometimes called *downgrading policies* [17, 40].

That experience is discussed in Section 2.3.

2.1 The Argument for a Reactive Language

Examples from data-use contracts, existing privacy policies, and U.S. regulations led us to identify four key attributes of a language for specifying use-based privacy policies:

Data-centric: ***A use-based privacy language must be able to associate policies with data.*** Since societal determinations about harmful uses are independent of individual preferences, use-based privacy policies are better coupled to the type of data than to the data subject. For example, a corporate privacy policy that states, "Click history may be used to personalize content" is a policy that applies to all click histories and does not depend on data subject preferences. Some use-based privacy policies do include authorizations that depend on user actions or user preferences settings. For example, HIPAA authorizes health care providers to share directory information with clergy members unless the data subject objects. Such policies can be expressed if a policy specification (associated with a value) is allowed to depend on system state.

Provenance-Dependence: ***A use-based privacy language must support provenance-dependent policies.*** During program execution, information flows from values to derived values. A use-based privacy regime must associate policies with those derived values. For example, social norms might preclude ads targeted according to date of birth while allowing ads to

depend on birthday ("Happy Birthday!"), even though birthday is derived from date of birth. Or, legal regulations might prohibit specific details in a health record from being used for medical research, but might allow research using statistics derived from collections of health records. A policy might state that contact information may be shared with third parties only after opt-in authorization is received from the data subject. Or advertising might be allowed based on a single HTTP request (i.e., re-marketing) but might be prohibited based on values derived from the entire browsing history (i.e., targeted advertising). We conclude that policies are best defined as sets of restrictions that depend on the sequence of operations by which the current value was derived, and these provenance-dependent authorizations must be propagated from initial values to derived values.

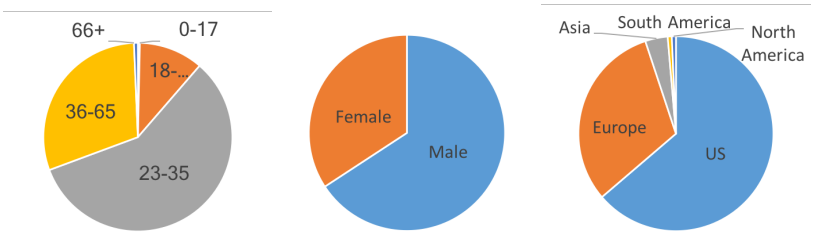
Restriction Type: ***A use-based privacy language must be able to express permissions, prohibitions, and obligations.*** Use-based privacy policies are often expressed as permitted uses or prohibited uses, e.g., "email address may be used to send notifications" or "email address may not be used to send promotional offers." A language that expresses only permitted or prohibited uses is likely to be inadequate, though, because use-based policies might also be violated when no action is taken. For example, "Credit card information can be shared with third parties, but remote copies must be deleted within 90 days" requires a means to express *obligations* [22, 31]—mandatory uses that must occur before a certain amount of time passes.

Policy Scope: ***A use-based privacy language must be able to express both sticky policies and local policies.*** Many use restrictions

are broadly applicable. For example, the policy "Date of birth may not be used to target ads" is probably not restricted to particular companies or jurisdictions. So a use-based privacy language must be able to support *sticky policies*[2] [45, 1]—policies that are associated with a value and that apply to all uses of this value as it flows through the system. But in some cases, restrictions might apply only within a specific context. For example, HIPAA imposes data use restrictions on health care operators in the United States. Any covered entity in the United States should associate these use-restrictions with data they receive or generate, but these restrictions do not apply to principals operating in other jurisdictions. So an expressive language should also admit *local policies*, which do not propagate use restrictions to third parties.

All four attributes above are satisfied by a *reactive* policy specification language—one that specifies not only a current set of restrictions for a value but also describes how those restrictions change after a sequence of operations are applied to the value. Reactive policy specifications define restrictions for a value and are therefore data-centric; reactive policy specifications can be associated with values as information flow labels. Derived values are synthesized by operations applied to a value; reactive policy specifications thus naturally express provenance-dependent authorizations. Obligations define authorizations that change with the passage of time; by viewing the passage of time as an operation applied to data, we can express obligations with reactive policy specifications. Finally, when reactive policy specifications are associated with values, the authorizations they express are sticky by default; but by viewing data transmission as an operation applied to data, we can also express non-sticky (i.e., local) policies with reactive policy specifications.

[2]Note that sticky policies, unlike *information flow labels*, are not necessarily inherited by derived values.



(a) Age (b) Gender (c) Nationality

Figure 2.1: Study subject demographics

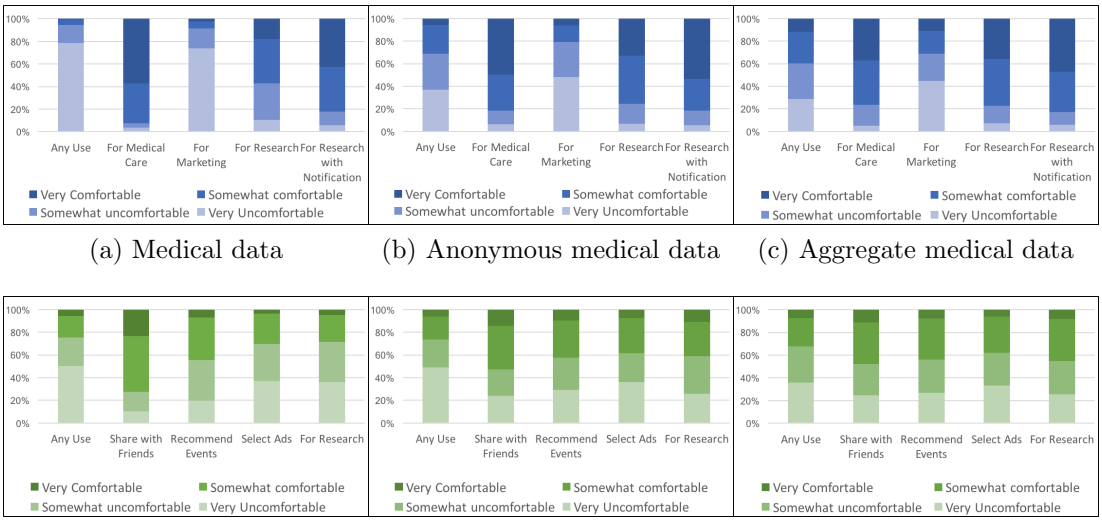
The connection between a reactive language and use-based privacy was motivated by reading various extant policies. But we wanted to affirm that a reactive language truly reflects priorities of people who use networked information systems. Note that we were not interested in individual preferences but rather in understanding societal definitions of harmful uses. So we conducted a user study in an attempt to explore whether reactive policies accurately capture collective preferences. We ran a study with 300 users using Amazon Mechanical Turk.[3] Respondents were limited to those with at least 50 approved HITs and at least a 90% approval rate; each respondent was rewarded with one dollar. The survey was posed as a sequence of multiple choice questions and a few free-form questions. There were three attention questions, one in each section of the survey[4]; responses that failed the attention questions were dropped. The study was run in three batches: a pilot study with 100 users, a follow-up study with 100 American respondents, and a follow-up study with 100 respondents who reside in the EU. Overall, respondents predominantly identified as American (63%) or European (32%) and slightly over half were male (65%). Age varied, but most subjects were working-age adults.

[3]Note that there are open questions regarding whether Mechanical Turk respondents are representative of society at large. We do not resolve those questions. Nonetheless, we view the results of this study, however imperfect, as evidence that social norms might be reactive.

[4]Each question asked, "Are you reading the questions and making an effort to answer them honestly?" The answer format matched the format of the surrounding questions: multiple choice in the first section, and free-form text in the other two sections.

	Coarsen	Anonymize	Aggregate	Passage of Time
Yes	170	191	179	83
Maybe	50	40	48	52
No	80	69	173	165

Figure 2.2: Factors influencing user data use preferences



(a) Medical data (b) Anonymous medical data (c) Aggregate medical data

(d) Social network data (e) Anon. social network data (f) Aggr. social network data

Figure 2.3: User preferences comfort with data use

The median completion time for the full survey was 7.9 minutes. The full survey is reproduced in Appendix A.

We first asked users whether they thought permitted uses might need to change when various operations are applied to data: 57% thought their preferred policy would change if their data were coarsened, 63% thought their data use preferences would change if the data were anonymized, and 60% thought their data use preferences would change if their data were aggregated with that of other users. In addition, 45% of users thought their preference would or maybe would change with the passage of time. These results, reproduced in Figure 2.2, reinforce a view that reactive policies are a natural match for representing societal preferences.

We further explored the relationship between collective user preferences and reactive policies by surveying respondents about their comfort with various uses of both raw data and data that had an operation applied (anonymized data or aggregated data). We asked each respondent to imagine an organization that stored and used their data (a health care provider or a social network) was defining a new data use policy, and we queried how comfortable the respondent would be to permit each of a set of specific proposed uses—each of which might be applied to either raw, anonymized, or aggregated data—using a four-point Likert scale: very uncomfortable, somewhat uncomfortable, somewhat comfortable, or very comfortable. The results are shown in Figure 2.3. How comfortable respondents were with medical data being used for research purposes depended on whether or not the use was preceded by a notification. Moreover, except for particular conditions that were deemed broadly acceptable under any circumstances (use for providing medical care, use for research with notification), reported comfort levels varied significantly between raw medical data (Figure 2.3a) and medical data that had been aggregated or anonymized (Figures 2.3b and 2.3c), also supporting our belief that reactive policies are a natural match for expressing societal preferences. In general, users were less comfortable with various proposed uses for social network posts than for medical data. However, some similar trends emerged. Users were significantly more likely to be comfortable with anonymized posts or aggregated posts being used to provide personalized recommendations, to select adds, or to conduct research (Figure 2.3e and Figure 2.3f vs. Figure 2.3d). Users were also more likely to allow aggregated information derived from their posts to be published publicly.

Overall, our survey results affirm that societal norms for harmful uses—as reflected by existing policies and regulations and by collective end user preferences—

depend on the history of events (environmental events and functions applied to data) that have occurred. This, in turn, supports the view that a reactive approach is well-suited for expressing use-based privacy.

2.2 A Reactive Language for Use-based Privacy

To further explore the feasibility of use-based privacy, we developed a language for specifying use-based privacy policies. *Avenance policies*[5] are predicates associated with values that specify whether a use is prohibited or allowed after a sequence of operations have been applied to that value. So, for example, an Avenance policy associated with a user's date of birth might not initially allow that value to be used for advertising, but after the year has been deleted, might allow the resulting value (the user's birthday) to be used for advertising.[6] Avenance policies are associated with sensitive values to form tagged values.

Avenance policies ρ are specified as conjunctions and disjunctions of *policy rules*

$$\rho := r \mid \rho \wedge \rho \mid \rho \vee \rho.$$

A policy rule r is represented as a privacy automaton—a finite state automaton that encodes history-dependent use-based authorizations. Formally, a policy rule is defined as a 5-tuple

$$r := (\mathsf{transType}, S, s_0, T, s_v)$$

where $\mathsf{transType}$ is the alphabet for transitions (i.e., the set of events in a history that might change the current set of authorized uses for a value), $S := \{s_0, \ldots, s_n\}$

[5]The term Avenance is new, but was derived from the French word *avenir*, meaning future or yet to occur; the etymology is analogous to that of provenance (from *provenir*).

[6]For practical reasons, we restrict Avenance policies to uses determined solely by the party currently holding the associated value.

is the set of states, s_0 is the initial state, T is the state-transition function

$$T: \ S \times \mathsf{transType} \rightarrow S,$$

and s_v is the violation state. Observe that, unlike standard finite-state automata, privacy automata do not explicitly define a set of accepting states; they instead specify a violation state s_v. A sequence of uses is *policy-compliant* if each use is authorized by the current state at the time that use occurs and if it never enters the violation state s_v.

A state s_i in a privacy automaton defines the set of permitted uses when the privacy automaton is in that state. This set of permitted uses is specified by conjunctions and disjunctions of *authorization triples*, predicates expressed as triples (I,E,P), where I identifies an invoking principal, E is some executable binary, and P denotes the purpose for executing E.

$$s := (\mathsf{I},\mathsf{E},\mathsf{P}) \mid s \wedge s \mid s \vee s$$

I may be defined as a single principal or may be a role, E may be specified by a binary hash or by a type drawn from a hierarchy of program labels, and P may be drawn from a hierarchy of purpose labels. *Compound components* I, E or P are constructed using unions and intersections.

$$\begin{aligned} \mathsf{I} &:= \mathsf{invoker} \mid \mathsf{I} \cap \mathsf{I} \mid \mathsf{I} \cup \mathsf{I} \\ \mathsf{E} &:= \mathsf{useType} \mid \mathsf{E} \cap \mathsf{E} \mid \mathsf{E} \cup \mathsf{E} \\ \mathsf{P} &:= \mathsf{purpose} \mid \mathsf{P} \cap \mathsf{P} \mid \mathsf{P} \cup \mathsf{P} \end{aligned}$$

Semantically, an authorization triple $(\mathsf{I}, \mathsf{E}, \mathsf{P})$ specifies a predicate that allows a use if it is in all three component sets; a state is interpreted as a predicate defined by conjunctions and disjunctions of authorization triples. So, in effect, we are equating authorization triples with uses.

Some simple policies can be expressed by authorization triples in a single-state privacy automata:

Example 2.1. *Data may be viewed by medical personnel for the purpose of providing counseling or medical care:*

$$\{(\mathsf{doctors} \cup \mathsf{nurses}), \mathsf{view}, (\mathsf{counseling} \cup \mathsf{medical\ care})\}$$

Example 2.2. *Data may be viewed by coaches and by researchers; researchers may use data to conduct research:*

$$\{(\mathsf{coach} \cup \mathsf{researcher}, \mathsf{view}, *) \vee (\mathsf{researcher}, *, \mathsf{research})\}$$

Authorized uses generate events when invoked; events that trigger automata state transitions are elements in the language transType. We consider two classes of events: *environmental events*—which modify the set of permitted uses for the associated value—and *synthesis events*—which generate new data values and trigger automata state transitions that specify the set of permitted uses for the derived values.

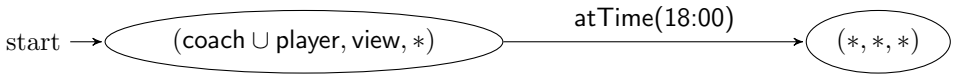
$$\mathsf{transType} := \mathsf{eEvent} \cup \mathsf{sEvent}$$

Note that the invocation of an authorized use might generate both an environmental and a synthesis event.

Environmental events are specified by the alphabet eEvent. Environmental events might be triggered by authorized uses, for example, the event sentToRemotePrincipal might be triggered by programs that send a copy of the value. Environmental events might also include *temporal events*—clock-triggered events that can be expressed either absolutely ($\mathsf{atTime}(t)$) or a relatively ($\mathsf{afterTime}(t)$)—or user actions (e.g., a change in a user's privacy settings). The mechanism responsi-

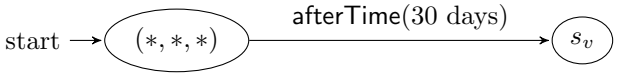
ble for enforcing policy compliance is responsible for updating the state of affected policy rules when environmental events occur.

Example 2.3. *Data may be viewed by coach and players; data becomes public after 18:00.*



Temporal events are handy for expressing obligations. To capture such obligations, we employ a violation automata state s_v. If a privacy automaton enters violation state s_v, then that policy (i.e., the obligation) has been violated.

Example 2.4. *Data must be deleted within 30 days.*



Synthesis events, drawn from the alphabet sEvent, are triggered by operations applied to a value that generate new data values. These events induce *policy derivation rules*: the policy associated with the output value of a synthesis event is the conjunction of the policies associated with the tagged values that influence the new value according to standard information flow rules; this includes implicit information flows. The current state of each privacy automata (i.e., policy rule) in the derived policy is determined by matching the synthesis event against the transitions in each of the input automata.

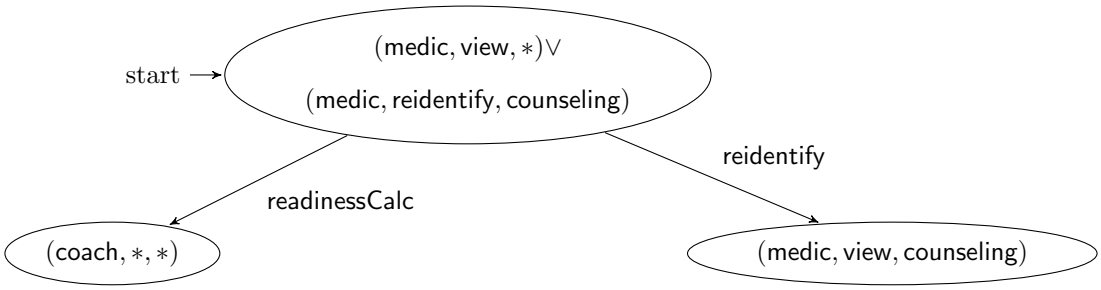
So, for example, if a value (e.g., anonymous mood data) is associated with the policy shown in Example 2.5 (below) and is then reidentified, the resulting derived data (i.e., the reidentified mood data) may be viewed by medics for the purpose of providing counseling. If the data used to reidentify the pseudo-anonymized mood

Authorization Triples:	States:	Policy Rules:
$\mathsf{I} := \mathsf{invoker} \mid \mathsf{I} \cap \mathsf{I} \mid \mathsf{I} \cup \mathsf{I}$	$s := (\mathsf{I},\mathsf{P},\mathsf{E}) \mid s \wedge s \mid s \vee s$	$r := (\mathsf{transType}, S, s_0, T, s_v)$
$\mathsf{P} := \mathsf{purpose} \mid \mathsf{P} \cap \mathsf{P} \mid \mathsf{P} \cup \mathsf{P}$	$S := \{s_1, \ldots, s_n\}$	
$\mathsf{E} := \mathsf{useType} \mid \mathsf{E} \cap \mathsf{E} \mid \mathsf{E} \cup \mathsf{E}$		
Transition Types:	Transitions:	Policies:
$\mathsf{transType} := \mathsf{eEvent} \cup \mathsf{sEvent}$	$T: \ S \times \mathsf{transType} \to S$	$\rho := r \mid \rho \wedge \rho \mid \rho \vee \rho$

Figure 2.4: Avenance Policy Syntax.

data (e.g., a mapping between pseudonyms and identities) was also associated with an Avenance policy, then the resulting identified mood data would be associated with the conjunction of two policies: one derived from the automaton associated with each of the two inputs.

Example 2.5. *Anonymous mood data may be viewed by medical personnel. Anonymous mood data may be re-identified and subsequently viewed by medical personnel for the purpose of providing counseling. Derived readiness score may be used by coaches.*



The syntax for Avenance policies is summarized in Figure 2.4. Observe that Avenance policies are constructed from sets of labels invoker, purpose, useType, eEvent, sEvent; these sets are defined by some namespace associated with the policy. We presume an agreed *interpretation* for each label—a legally-binding definition of the programs comprising that type (e.g., a specific natural-language explanation). Different namespaces could be defined by individual data stores,

service providers, or established nonprofit organizations. One example namespace is given in Example 2.6.

Example 2.6. *PMSys is a mobile and web-based application developed at the University of Tromsø that performs physiological evaluation and training-load personalization for soccer players. PMSys collects data about player mood, sleep patterns, physical fitness, and injuries. These data are subject to data-use restrictions derived both from the data-use contract signed by the players (who all are members of elite clubs and national teams in Norway, Sweden, and Denmark). The following namespace was defined to support policies for the PMSys soccer application:*

$$
\begin{aligned}
\mathsf{invoker} &:= \{\mathsf{medical}, \mathsf{coach}, \mathsf{player}\} \\
\mathsf{purpose} &:= \{\mathsf{prevention}, \mathsf{diagnosis}, \mathsf{care}, \mathsf{intervention}\} \\
\mathsf{useType} &:= \{\mathsf{view}, \mathsf{sendTo}(\cdot), \mathsf{delete}, \mathsf{average}, \mathsf{pseudonoymize}, \mathsf{reidentify}, \\
&\qquad \mathsf{readinessCalc}, \mathsf{rosterCalc}\} \\
\mathsf{eEvent} &:= \{\mathsf{atTime}(\cdot), \mathsf{afterTime}(\cdot)\} \\
\mathsf{sEvent} &:= \{\mathsf{average}, \mathsf{pseudonoymize}, \mathsf{reidentify}, \mathsf{readinessCalc}, \mathsf{rosterCalc}\}
\end{aligned}
$$

2.3 Evaluating Avenance Policy Expressiveness

Avenance policies have utility only if they are able to describe use-based privacy policies arising in practice. We selected two real-world policies to use for evaluating the expressiveness of the Avenance language. Medical care is one field that could benefit from broader data sharing if (and perhaps only if) strong privacy guarantees can be maintained, so we selected the United States federal regulation on privacy for health data, the Health Information Portability and Accountability Act (HIPAA), as one example policy. Recent headlines, including reported uses of Facebook data by Cambridge Analytica and other business partners, have high-

lighted the wealth of personal data collected and shared by social networks, and the demand for stronger privacy controls for this data. So we selected Facebook's site privacy policy [25] as our second example policy. We believe that each example policy is representative of a class of data use policies (privacy regulations and site privacy policies, respectively) that constitute common sources of use-based policies. For each, we determined whether and how the imposed use-restrictions might be expressed as Avenance policies. Success with these example applications increases our confidence in the broad applicability of our reactive approach to use-based privacy.

2.3.1 HIPAA

HIPAA regulates members of the health care industry. In addition to defining rules for information storage and security, it imposes limitations on how health providers or *covered entities* may use and disclose personal health information.

To encode HIPAA's data-use rules, we use a variant of Semantic Parameterization [8]. For each rule, we identify five properties constraining a use:

1. The *object* is the data to which the use restriction applies.
2. The *invoker* is the principal that invokes the use.
3. The *purpose* is the goal of the activity.
4. The *action* is the type of use covered by the rule.
5. The *condition* is a Boolean expression indicating when the rule applies.

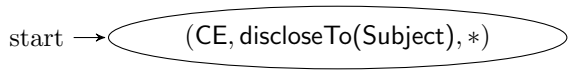
We analyzed §164.502-§164.528 of the HIPAA Privacy Rule—the sections describing restrictions on how data may be used—and extracted 95 data-use tuples.

We then manually analyzed each of the resulting data-use tuples, and we formalized each tuple as an Avenance policy rule. The full encoding is given in Appendix B.

Inspecting the resulting policies, we observed that invoker, purpose, and action defined an authorization triple $(\mathsf{I}, \mathsf{E}, \mathsf{P})$ that could be expressed in the Avenance language. Some data-use rules referred to a particular action (e.g., Example 2.7), but much of HIPAA describes restrictions on use according to purpose (e.g., Example 2.8); the Avenance language allows us to express both types of restriction.

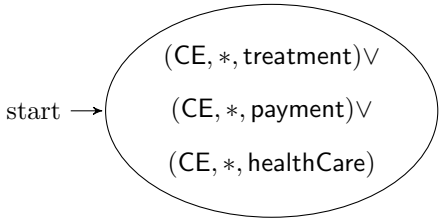
Example 2.7. HIPAA §164.502(a)(1)(i): *A covered entity is permitted to disclose protected health information to the individual.*

Object	Invoker	Purpose	Action	Condition
PHI	CE	$*$	discloseTo(Subject)	



Example 2.8. HIPAA §164.506(c)(1): *A covered entity may use or disclose protected health information (PHI) for its own treatment, payment, or health care operations.*

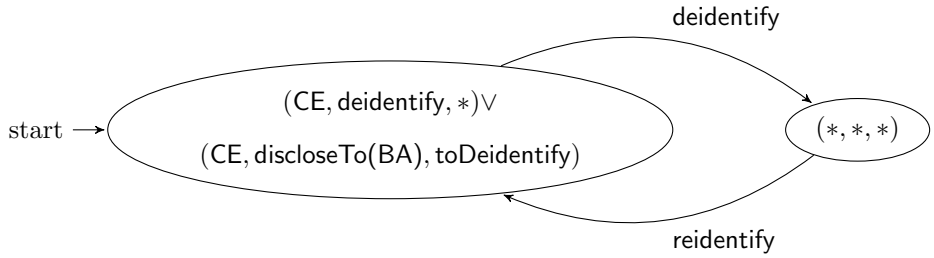
Object	Invoker	Purpose	Action	Condition
PHI	CE	treatment	$*$	
PHI	CE	payment	$*$	
PHI	CE	healthCare	$*$	



Some HIPAA privacy rules refer to particular classes of objects objects (e.g., de-identified health information); these define how data may be used after specific transformations have occurred.

Example 2.9. HIPAA §164.502(d): *(1)A covered entity may use protected health information to create information that is not individually identifiable health information or disclose protected health information only to a business associate for such purpose, whether or not the de-identified information is to be used by the covered entity. (2) Health information that meets the standard and implementation specifications for deidentification under §164.514(a) and (b) is considered not to be individually identifiable health information, i.e., de-identified. The requirements of this subpart do not apply to information that has been de-identified in accordance with the applicable requirements of §164.514, provided that: (i) [...] and (ii) If de-identified information is re-identified, a covered entity may use or disclose such re-identified information only as permitted or required by this subpart.*

Object	Invoker	Purpose	Action	Condition
PHI	CE	$*$	deidentify	
PHI	CE	toDeidentify	discloseTo(BA)	
de-identified HI	$*$	$*$	$*$	

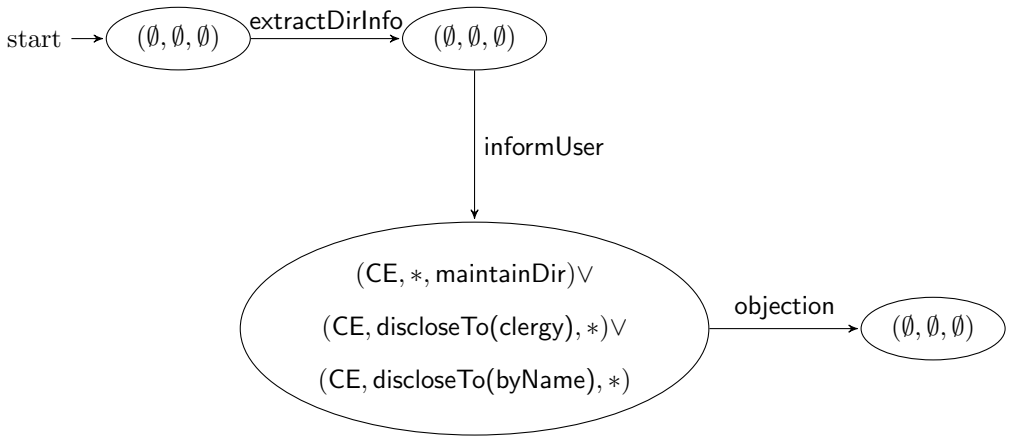


Other HIPAA rules specify conditions under which that rule applies. These conditions would be encoded in an Avenance policy using environmental events.

In the following example, specific information has additional permissions, but only as long as the user (i.e., the subject of the data) is aware of the possible use and does not object. To encode this data-use rule, we employ two environmental events—informUser and userObjection—to specify how the authorization changes after the data subject has been informed of the possible use and after the data subject registers an objection to those uses.

Example 2.10. HIPAA §164.510(a)(1): *Except when an objection is expressed in accordance with paragraph (a)(2) of this section, a covered health care provider may: (i) Use the following protected health information to maintain a directory of individuals in its facility: (A) The individual's name; (B) The individual's location in the covered health care provider's facility; (C) The individual's condition described in general terms that does not communicate specific medical information about the individual; and (D) The individual's religious affiliation; and (ii) Disclose for directory purposes such information: (A) To members of the clergy; or (B) to other persons who ask for the individual by name. (2) Opportunity to object. A covered health care provider must inform an individual of the protected health information that it may include in a directory and the persons to whom it may disclose such information (including disclosures to clergy of information regarding religious affiliation) and provide the individual with the opportunity to restrict or prohibit some or all of the uses or disclosures permitted by paragraph (a)(1) of this section.*

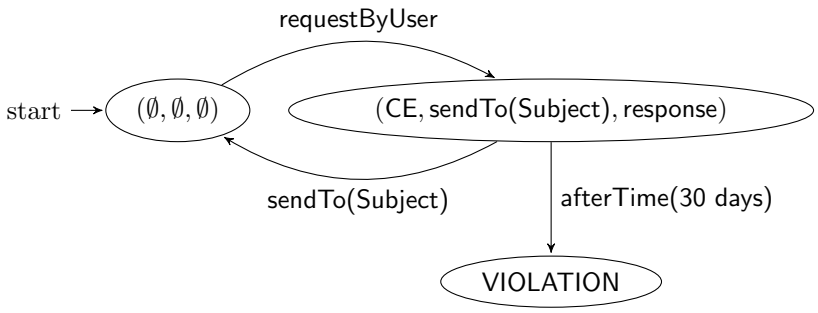
Object	Invoker	Purpose	Action	Condition
directory info	CE	maintainDir	$*$	after informUser unless objection
directory info	CE	$*$	discloseTo(clergy)	after informUser unless objection
directory info	CE	$*$	discloseTo(byName)	after informUser unless objection



Finally, some HIPAA privacy rules impose obligations rather than expressing permissions. These rules are expressed in the Avenance language by using temporal transitions. Actions that fulfill obligations are environmental events, which trigger a state transition.

Example 2.11. HIPAA §164.524(b): *(1) The covered entity must permit an individual to request access to inspect or to obtain a copy of the protected health information about the individual that is maintained in a designated record set. (2)(i) The covered entity must act on a request for access no later than 30 days after receipt of the request as follows.*

Object	Invoker	Purpose	Action	Condition
PHI	CE	response	SendTo(Subject)	



Because we were able to express all 95 identified data use rules as Avenance policy rules, our experience with HIPAA confirms that types of data-use restrictions defined by the HIPAA Privacy Rule can be expressed as rules in the Avenance policy language.

Five HIPAA data-use rules, however, illustrate what might be termed *second-order use restrictions*. For example, §164.508, which deals with user authorizations (and exceptions to the authorization requirement), includes the statement that covered entities may use protected health information for any use explicitly authorized by the user.

Example 2.12. HIPAA §164.508(a)(1): *Except as otherwise permitted or required by this subchapter, a covered entity may not use or disclose protected health information without an authorization that is valid under this section. When a covered entity obtains or receives a valid authorization for its use or disclosure of protected health information, such use or disclosure must be consistent with such authorization.*

This privacy rule can be expressed in the Avenance language using an unin-

terpreted label authorizedUses. However, we do not consider such a formulation to be useful for policy enforcement. Instead, we would expect to handle second-order rules simply by adding additional policy rules (corresponding to the authorized uses) to the Avenance policy at the time an authorization is received.

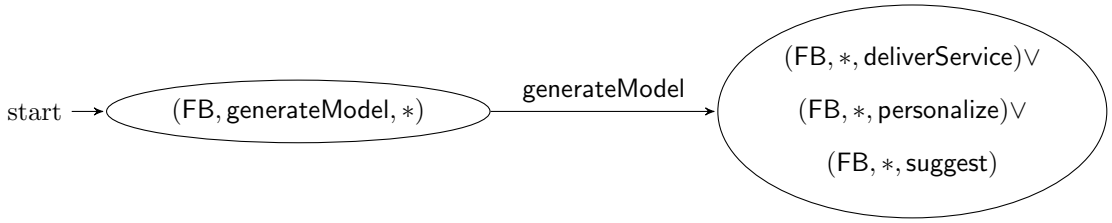
2.3.2 Facebook Privacy Policy

For our second case study, we selected of Facebook's Data Use Policy [25], which states how Facebook uses information it collects. Facebook is a widely used service provider; the terms of its data use policy are typical for social networking sites. The ability to express Facebook's data use policy is therefore a positive indicator for the Avenance approach.

We performed a detailed analysis of Facebook's data use policy. Adopting the standard legal interpretation, we view any stated use as a permission. We employ the same methodology as we used to analyze HIPAA: first code each sentence of the data use policy using our five-fold attribute classification (this results in a set of 38 data use tuples) then formulate each of these tuples as an Avenance policy rule. An example data use rule from Facebook's data use policy is given in Example 2.13; the full data use policy is described in Appendix C.

Example 2.13. *We are able to deliver our Services, personalize content, and make suggestions for you by using this information to understand how you use and interact with our Services and the people or things you're connected to and interested in on and off our Services.*

Object	Invoker	Purpose	Action	Condition
PI	FB	$*$	generateModel	
user model	FB	deliverService	$*$	
user model	FB	personalize	$*$	
user model	FB	suggest	$*$	



Facebook's data use policy describes permitted behaviors both in terms of particular functions (e.g., generating a user model) and general purposes that might motivate a variety of different functions (e.g., delivering services). Many rules depend on state transitions triggered by both synthesis (e.g., user model generation, anonymization, aggregation) and environmental events (e.g., changes in user settings). Facebook promises to delete posts after an account is deleted, defining an obligation. Note, however, that Facebook's data use policy does not contain any second-order rules. The published policy does modify the set of valid authorizations when a user updates preference settings; however this set of settings is a finite, pre-defined set that can be expressed by our first-order semantics.

CHAPTER 3

ENFORCEMENT WITH BENIGN SERVICE PROVIDERS

In this chapter, we consider the problem of enforcing compliance with use-based privacy policies for benign service providers.

> ***Benign Service Providers.*** Service providers are presumed to have non-technical incentives to comply with all relevant policies (e.g., concern for reputation or legal consequences). They might violate a policy due to a mistake or a programmer error, but they are not actively malicious.

Here, it suffices to provide tools that facilitate policy compliance for well-intentioned, but imperfect, service providers. We propose to facilitate policy compliance by (1) separating the policy from the code, thereby improving modularity and extensibility and (2) introducing an automated compliance checker. With the Avenance language, policy experts, working with a corporate legal team, can encode relevant regulations and data-use contracts as Avenance policies. Independently, developers can annotate application code with applicable use types. There is no automatic verification of annotations. However, by assuming benign service providers, we are presuming that service providers perform a manual code review to ensure annotation correctness. The automated compliance checker then enforces use-based privacy policies on the basis of those annotations. This enforcement can either be *prevention-based*—the compliance checker blocks unauthorized uses—or *detection-based*—the compliance checker creates an audit log that records all uses of tagged values. The prevention-based approach offers improved performance, but it is unable to prevent all policy violations; it cannot prevent violations of obligations, and enforcement might be imprecise if application code is not correctly annotated. The detection-based approach relies on *post-facto* auditing and deter-

rence through accountability—such monitors have been shown to reliably detect violations of real-world policies (e.g., HIPAA) [26].

In this chapter, we present two approaches to automatically checking policy compliance in the presence of benign adversaries: LoNet, which modifies a file system to implement policy enforcement at the granularity of operating system objects (e.g., processes and files) and Tir, which provides a library for augmenting programs with language-level inline monitoring.

3.1 LoNet

LoNet is a runtime system deployed inside a machine virtualization container (e.g., VirtualBox [69] or Docker [21]). It implements a reference monitor that intercepts data accesses to ensure compliance with Avenance policies. LoNet adopts the file system as its primary interface.[1] LoNet stores all values in files, and files are organized in hierarchical directories. Each file can be associated with a policy file that contains an Avenance policy. LoNet intercepts system calls that read or write files and mediates those accesses to enforce compliance with the Avenance policy defined by the associated policy file.

3.1.1 Policy Expression

LoNet supports a simple but expressive subset of the Avenance language. A LoNet policy file encodes the authorizations of a single state in a privacy automaton. Syn-

[1] Although we do not directly support databases or structured files, fine-grained policies could be implemented by employing an appropriate file-based data model for databases or by otherwise subdividing data into many small files.

```
import errno, time, os
from fuse import FuseOSError

# setup by LoNET: path

if (time() - lstat(path).st_ctime) > 86400:
    raise FuseOSError(EACCES)
```

Figure 3.1: The meta-code permiting access within 24 hours.

tactically, it contains contains three components: (1) a list of invokers I permitted to use the associated data file in any way (this correspond to authorization triples of the form $(\mathsf{I}, *, *)$), (2) a mapping from executable types E to policy files specifying which policy file to associate with a derived file synthesized by a program of type E (which both induces authorization triples of the form $(*, \mathsf{E}, *)$ and defines automata state transitions triggered by synthesis events E), and (3) a mapping from executable types E to code snippets or *meta-code* that should be run when programs of type E are invoked. Meta-code can be used to specify automata state transitions triggered by an environmental event eEvent. For example, to impose an obligation that a particular file may be viewed only for the next 24 hours, the policy could use the meta-code shown in Figure 3.1, which throws an error if the time limit has expired.

Consider the example set of policy files is given in Figure 3.2: The policy `/pol/priv` from Figure 3.2a defines a privacy automata state with authorization $(\mathsf{owner}, *, *) \vee (*, \mathsf{download}, *)$ and with state transition $\mathsf{download} \mapsto$ `/pol/raw`. If that policy file were associated with a credential file containing a user's Fitbit OAuth2 token, then LoNet would intercept system calls that access the credential file and enforce that only a container run by the credential owner may invoke a program of type view on that file (since any program run by invoker owner may access the file, but only programs of type $\mathsf{download}$ may access the file when

```
[permissions]
roles = {owner}

[transitions]
download=/pol/raw
```

(a) /pol/priv

```
[permissions]
roles = {medic}

[transitions]
smoothing=/pol/smoothed
desensitize=/pol/desens

[metacode]
view=/code/24hr
```

(b) /pol/raw

```
[permissions]
roles = {coach}

[transitions]
desensitize=/pol/pub
```

(c) /pol/smoothed

```
[permissions]
roles = {medic}

[transitions]
smoothing=/pol/pub
```

(d) /pol/desens

```
[permissions]
roles = {*}
```

(e) /pol/pub

Figure 3.2: A set of example LoNet policy files.

invoked by any other invoker). Files derived from the credential file inherit a policy file determined by the policy file associated with the credential file and the type of program that created them, following the policy derivation rule described in Chapter 2. For example, if a medic (who cannot view the credential file) runs a program with executable type download that uses the credentials file to download raw data from the Fitbit data store (which they are authorized to do), then the downloaded data file will be associated with the policy file `/pol/raw` defined in Figure 3.2b. The policy on the downloaded data enables the medic to initially read the output data. However, this policy specifies that the `/code/24hr` meta-code should run each time view is invoked. The meta-code, shown in Figure 3.1, compares the file creation time with the current time and returns the `EACCESS` error if the difference is more than 24 hours. The Avenance privacy automata

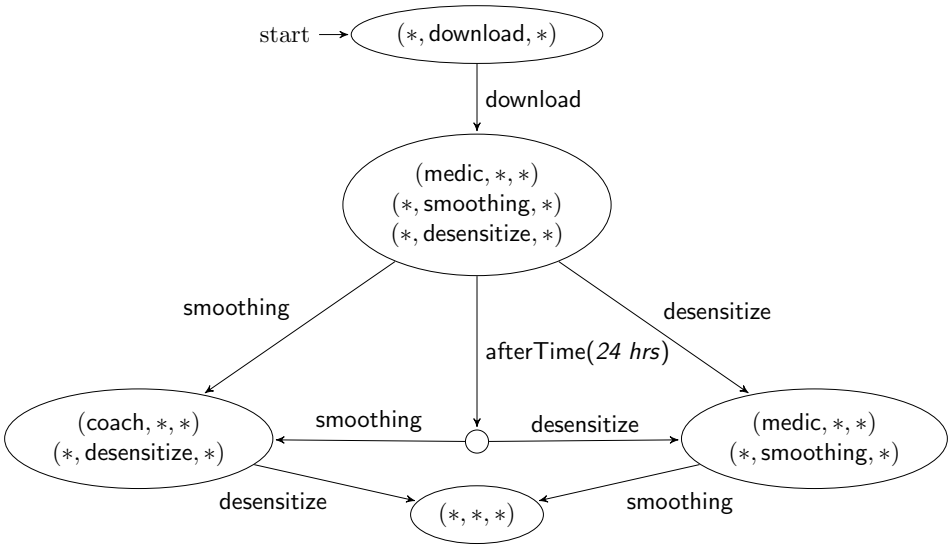


Figure 3.3: A graphic depiction of the policy automaton described by the policy files in Figure 3.2.

corresponding to this example is shown in Figure 3.3.

LoNet supports both mandatory and discretionary policies. A policy for a file can be defined by the file's owner at the time the file is created. Policies can also be added subsequently, by a data processor. In addition, any newly created file inherits policies from files on which the content of the new file depends, according to the derivation rules defined by the Avenance language. Using LoNet, a user can therefore safely share a file (and the associated policy) with other users; LoNet will permit the receiver to use the file contents only in a manner authorized by the associated Avenance policy.

3.1.2 Implementation

We implemented a prototype of LoNet using the FUSE user-level file system library (available in Linux 2.6 and Linux 3 kernels) in combination with VirtualBox containers. The LoNet daemon process runs as a privileged system user inside the

container; it has full access to a hidden source file-system directory. The source directory is exported to a known mount point accessible to system processes. The authorizations of those system processes are controlled by LoNet. The LoNet daemon intercepts all file system calls to the visible mount point and mediates file access, invokes meta-code execution, and invokes policy derivation routines to enforce and propagate the security policies attached to files.

Invoker types are implemented as roles. Users acting in a role interact with protected objects within the LoNet sandbox through traditional file system abstraction and tools. Authorized users (e.g., data owners and data processors) may also associate additional policies with files using the extended file attributes feature available in modern Linux kernels and filesystems. To associate a policy with a file, a principal uses the `setfattr` command. For example, an athlete can express the policy from Figure 3.3 with a file `credentials` using the command

```
#> setfattr -n policy -v /pol/priv credentials
```

LoNet reads the extended file attribute `policy` when mediating system calls that access a data file and enforces the Avenance policy defined in the named policy file.

As described in Section 3.1.1, an Avenance policy in LoNet might depend both on the invoker type I—the role of the user that invokes the LoNet container—and on the executable type E of the program that is invoked. To support policies that depend on the invoker's role, FUSE needs to inform LoNet about which principal is accessing the file system, and in which role. Unfortunately, the FUSE interface only provides Linux process-id, user-id, and group-id of the calling process. To circumvent this problem, LoNet requires that user certificates are passed as part

of the file name itself, as the first component of a Linux path name.

To support policies that depend on the executable that is invoked, LoNet defines executable types E as pairs comprising an identifier for the program (either the SHA-256 hash of program binary or a `exe_path` value to a trusted binary location) and an expression that restricts arguments of that program. Restrictions on arguments enable LoNet to map programs expressed in scripting languages, like Python or R, based on both the binary and the script file being loaded. Trusted programs that correspond to a particular E are specified in a configuration file, as illustrated in Figure 3.4. When a program attempts to access a file, LoNet maps the process-id to the appropriate executable type by (1) reading and hashing the executable binary for the process-id as provided through `/proc/{pid}/exe`; (2) reading and parsing the program arguments as provided through `/proc/{pid}/cmdline`; and (3) matching the hash of the executable and its arguments to an executable type E as specified in the executable type configuration file. To improve performance, LoNet maintains a cache of recently seen pids and their executable types, which avoids some of the cost related to the mapping of process-ids to executable types.

To implement the Avenance policy derivation rule while supporting automation tools and interactive programs—like GNU Make or bash-shells—that might invoke multiple other sub-programs, LoNet implements *sessions*. Each program is run in the context of a session, and each session is associated with a principal. Sessions are used to implement policy propagation; each time a program accesses a data file, the session is tainted with the policy file associated with that data file. The policy for an output file is the conjunction of the policy files in the session taint. By propagating taint exclusively within a session, LoNet limits over-tainting.

A program initiates a new session by reading from dedicated file `/newsession`—

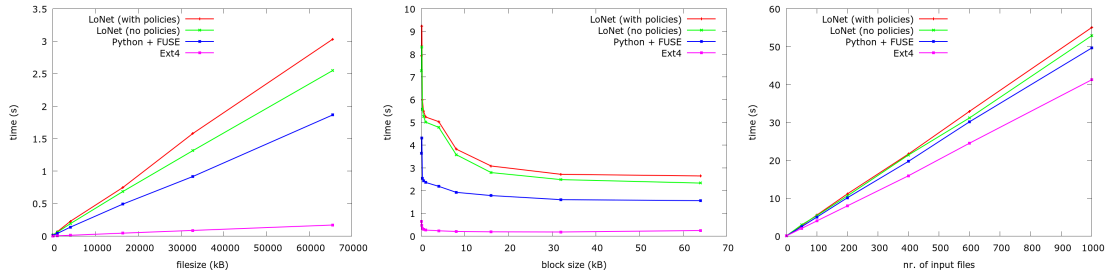
```
[cat]
exe_path = /bin/cat
useType = publish

[Analyze]
exe=e671...
options_match = -B -d -E -h -i -O -OO ...
options_havearg = -m -Q -W -c
arg0_type = hash
arg0 = 0x[...]17011
ttype=desensitize

[Download]
exe=e671...
options_match = -B -d -E -h -i -O -OO ...
options_havearg = -m -Q -W -c
arg0_type = path
arg0 = /code/download
ttype=project
```

Figure 3.4: Example executable type specification file.

which returns an unique 32-byte session identifier—and setting the `SESSION_ID` environment variable to the returned session identifier prior to the program's first file operation. Programs that do not explicitly specify a session identifier are mapped to a default `NULL` session; programs run in the context of the default `NULL` session might be associated with unnecessary meta-code.

3.1.3 Evaluation

To quantify the overhead of the policy-enforcing mechanism in LoNet, we evaluated a set of micro-benchmarks. We ran all experiments on a Lenovo Thinkpad T430s with an Intel Core i7-3520M CPU with four cores, each running at 2.90 GHz; 12 GB of main memory; and a 500 GB Samsung 840 EVO SSD drive. This hardware is



(a) Copy performance with file-size. (b) Copy performance with blocksize. (c) File aggregation performance.

Figure 3.5: Performance results for the LoNet prototype.

representative of what our expected end-user would have available. The machine runs 64-bit Ubuntu 13.04, and we use Anaconda Python 3.4 to run LoNet.

First, we measure how IO performance is affected by our meta-code propagation and policy transition mechanisms by measuring the time it takes a Python script to copy data between two files within the LoNet container. The script reads and writes using 32 kB blocks; the initial files contain random bytes. Figure 3.5a shows the observed overhead of LoNet on files with and without meta-code policies for file sizes ranging from 1 kB to 64 MB. For comparison, we also run the experiment directly on the underlying Ext4 filesystem, and on a minimal Python-based FUSE filesystem (Python + FUSE), which only passes invoked operations to the underlying filesystem. Each experiment is repeated 10 times; the figure reports the mean. As shown in Figure 3.5a, there is significant overhead associated with LoNet.

Figure 3.5b reports how block size affects our copy experiment. We set the file size to 64 MB and vary the block size between 64 B and 64 kB. As expected, the increase in IO operations due to a small block size adversely impacts LoNet's performance.

Finally, we evaluate how LoNet performs when aggregating a large number of files. For this, we implement a Python script that consecutively reads and computes the SHA-256 hash for a given set of files. Figure 3.5c shows the time it take to aggregate between 1 and 1000 files. Each test file contains 10 MB of random data. The block size is set to 16 kB. As shown in Figure 3.5c, computational intensive workloads are less affected by the IO overhead of LoNet.

3.2 Tir

LoNet suffers from three weaknesses. First, policy enforcement in LoNet imposes a significant performance cost, primarily by relying on a userspace filesystem (FUSE) and an interpreted language (Python). Second, LoNet was designed to track and enforce policies at the granularity of a file; fine-grained tracking and enforcement is not supported. Third, it does not support the full Avenance language. Therefore we designed and built a second system called Tir[2] that mitigates those shortcomings; Tir is a middleware layer that enforces Avenance policies at the granularity of individual values.

3.2.1 System Design

We designed an inline monitoring library that enables application developers to annotate regions of code that correspond to uses and to augment existing or future applications with an inline monitor that implements automated compliance checking. The library supports both prevention-based and detection-based enforcement;

[2]Tir is the Armenian god of learning and wisdom, who is responsible for recording men's actions, both good and bad.

polstore * init_polstore(char *l)	Initialize a polstore with logfile name l (may be `NULL` for prevention-based enforcement).
int store_policy(polstore *s, void *v, char *p)	Create a polstore entry in s for the value at location v and associate it with policy serialized as p.
int delete_policy(polstore *s, void *v)	Delete the entry associated with v from polstore s.
pol * retrieve_policy(polstore *s, void *v)	Return the policy from s associated with the value at location v.
int trans(polstore *s, char *i, char *p, char *t, int n, void *ins[], void *o)	Use the n values ins[] for use (i, p, t), where t is a transitions type, and associated the derived policy with the output stored in o.
void change_use(polstore *s, char *i, char *p, char *e, int b)	If $b = 0$ remove, add use (i, p, e) to the set of current uses for polstore s.
void *use(polstore *s, void *v)	Use the value v for the current use(s) defined in polstore s.
int check_policy(polstore *s, void *v, char *i, char *p, char *e)	Return a boolean indicating wither the use (i, p, e) is currently permitted by the policy associated with v.

Figure 3.6: Monitoring API for our inline monitor implementation.

the enforcement mode can be configured with a compiler flag. The API provided by the inline monitoring library is given in Figure 3.6; the behavior of each API call is described below.

The API call init_polstore initializes the monitor and creates a *policy store*, which will store tagged values; the policy store uses the memory address as an identifier for a value and maps addresses to policies. Tagged values can subsequently be added or removed from the policy store using API calls store_policy and delete_policy; the policy for a tagged value can be retrieved using the API call retrieve_policy. When the API call trans is invoked, the monitor automatically computes policies for the derived (output) value o based on the policies of the input values ins[] and the declared executable type $\mathsf{E} = t$ of the function that generates

the derived value; it then adds the derived tagged value to the policy store. If the library is configured for prevention-based enforcement, all API calls that modify the policy store append a log entry to the audit log maintained by the monitor.

Code snippets that correspond to particular executable types E are labeled using change_use to mark the beginning and end of code segments that implement a particular use. If the library is configured for prevention-based enforcement, this API call updates the list of current executable types; if the library is configured for detection-based enforcement, this API call appends a log entry to the audit log. When a tagged value is used, that use should be accompanied with a monitor call use that will enforce policy compliance. If the library is configured for prevention-based enforcement, this monitor call replaces the tagged value with a `NULL` pointer whenever the use is not authorized; authorization decisions are determined by the current set of executable types. If the library is configured for detection-based enforcement, this monitor call appends a log entry to the audit log for later inspection.

The inline monitor also includes a check_policy call; a policy-compliant application that uses the inline monitor in prevention mode is expected to call check_policy immediately prior to any call to use and only proceed if the use is authorized.

3.2.2 Implementation

We implemented Tir as a C library `libav` with 2701 lines of code [4]. Syntactically, Avenance policies are `json` encodings of lists of privacy automata; these lists are interpreted as conjunctions of policy rules. Header file `av.h` defines the public interface for Tir.

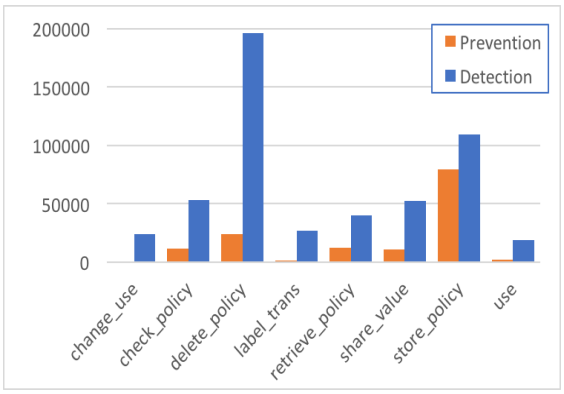


Figure 3.7: Performance of inline monitoring library calls.

3.2.3 Evaluation

To predict the performance of an inline monitor implemented with our library, we ran a series of microbenchmarks that evaluate the costs of various library calls. We ran our experiments on an OptiPlex 9020 with an Intel Core i7-4790 3.6 GHz CPU and 16GB of memory running Ubuntu 14.04 LTS (kernel version 3.13.0). The average latency of 1000 experiments is shown in Figure 3.7. Higher latency imposed by the detection-based monitor can be attributed to the cost of writing log entries to disk.

To evaluate the effect of automatic compliance-checking on application performance, we ported the PMSys application from Example 2.6—which requests raw values about players, computes the aggregate statistics, and then displays the results to the team coach—to run with inline monitoring. PMSys is implemented as a web application with 4300 lines of HTML, 140,000 of Javascript, and 10,000 lines of CSS. To enable compatibility with our inline monitoring library, we ported the key data analytics functionality to run in 1000 lines of C/C++. To annotate our code for inline monitoring, we needed to add 23 calls to the monitoring API. The effect on end-to-end latency of the application was negligible; performance of

the monitored code was within the error margin of our baseline experiment. Tir therefore offers significant performance improvement relative to LoNet at the cost of introducing a burden on application developers—programmers must implement their code with calls to the inlined monitor.

CHAPTER 4

ENFORCEMENT WITH ADVERSARIAL SERVICE PROVIDERS

Tools like LoNet and Tir are designed to enable service providers to ensure that they comply with all relevant use-based privacy policies. They are not, however, designed to enforce compliance if a service provider does not cooperate. An adversarial service provider can bypass LoNet by manipulating files outside of the LoNet filesystem or simply by not running LoNet at all. An adversarial service provider can bypass Tir by mislabeling a use type (e.g., annotating a disallowed use as a different permitted use) or omitting annotations when a use occurs (thereby bypassing the policy checks). In either case, there is also no way to ensure or check whether a particular service provider is (correctly) deploying these systems.

In this chapter, we describe our investigation of the feasibility of enforcing Avenance policies in the presence of adversarial service providers. We consider two possible threat models:

> ***Accountable Service Providers.*** Service providers are rational principals that act to optimize some utility function; they might knowingly violate use-based privacy policies under certain circumstances, for example, to increase profits. The utility function gives significant negative weight to being detected in a policy violation, so an accountable service provider will not run code that results in a policy violation that some auditor might detect. It suffices to detect violations in order to guarantee policy compliance by accountable service providers.
>
> ***Malicious Service Providers.*** Service providers here might knowingly violate use-based privacy by running code that results in a policy

violation—even if that violation might be detected. Such behavior must be prevented. A monitor that implements prevention is needed to enforce policy compliance by malicious service providers.

Accountable service providers are the appropriate threat model if service providers are subject to legal consequences or negative publicity. In other cases (e.g., if service providers can't be reliably identified or if they are irrational), service providers might not conform to the defining assumptions for accountable service providers and should instead be considered malicious.

To ensure policy compliance in the presence of accountable or malicious service providers, use-based privacy must be enforced whenever a service provider uses a value. As in the benign case, this can be accomplished by either prevention or deterrence through accountability. Both approaches involve monitoring accesses, and both require the monitor to be trusted to enforce that policy. To guarantee that the monitor is trustworthy when service providers are not benign, however, we need some means for monitoring behavior by service providers. Existing approaches to monitoring focus exclusively on read/write access control [41, 10] or on systems under the control of a single trusted authority [53, 57] and, thus, are unsuited for enforcing use-based privacy in distributed systems.

Recent developments in trusted hardware—e.g., Intel's Software Guard Extensions (SGX) [19]—offer a new basis for placing trust in a monitor or other program. Using SGX, an untrusted principal can provide a remotely-authenticatable proof or *quote* which attests that some program is running or has produced a given output. In this chapter, we investigate the feasibility of using Intel's SGX hardware as a root of trust, and we explore how we might leverage that root to implement

use-based privacy. An overview of relevant SGX features is given in Section 4.1.

We explore three possible architectures for enforcing use-based privacy in distributed systems with adversarial service providers. In our *source-based monitoring* architecture, *policy providers*[1]—trusted principals responsible for storing values and associating values with policies—run the monitors. Applications request values from policy providers; only those applications that provide appropriate credentials (e.g., SGX quotes) can gain access to sensitive values. The source-based monitoring architecture provides strong security guarantees. It is also easily deployed; application developers do not need to handle or interpret policies, and enforcement is compatible with legacy applications. However, this architecture exhibits poor performance for many applications. The architecture is described and evaluated in Section 4.2.

The performance limitations of source-based monitoring lead us to consider an alternative architecture called *delegated monitoring*, which improves throughput and reduces latency by locating the monitor at service providers. Delegated monitors act as proxies for local applications and use SGX quotes to prove to a policy provider that they are instances of a valid monitor; local applications use SGX to locally authenticate with the delegated monitor in order to gain access to data that is limited by policy to particular uses. The architecture provides the same strong security guarantees, while demonstrating significant performance improvements over source-based monitoring. However, delegated monitoring is less easily deployed than source-based monitoring, because service providers must run a delegated monitor and because local applications must interact with that monitor and store cached policies. This architecture is described in Section 4.3.

[1]Chapter 5 discusses the role of policy providers in our ecosystem, and describes the design and implementation of one possible policy provider.

The primary shortcoming of the delegated monitoring architecture is noticeable latency overhead for applications that handle lots of data or that enforce policy for fine-grained data. To eliminate this overhead, we consider a final architecture, *inline monitoring*, which inlines monitoring code directly into a monitored service provider application. This final architecture offers the best performance, particularly for applications that handle lots of data or fine-grained policies. However, the architecture (like other inline monitors, e.g., Tir) imposes a burden on application developers and, moreover, this approach is only able to guarantee policy compliance in an attenuated threat model. This architecture is described in Section 4.4.

4.1 An Overview of Intel SGX

Intel's Secure Guard Extensions (SGX) are an extension to the Intel x86 instruction set architecture. SGX uses chip-specific *hardware keys* to enable the construction of secure execution containers called *enclaves*; each enclave is isolated and supports data sealing, local attestation, and remote attestation.

Enclave Isolation. SGX enclaves provide confidentiality[2] and integrity for programs (and their data) running inside the enclave. This isolation is enforced by *processor reserved memory* set aside during boot and accessible only to SGX microcode and programs running within enclaves; the memory is partitioned into 4k pages, which are collectively referred to as the *enclave page cache* (EPC). Pages in the EPC are exclusively associated with a particular enclave and can only be accessed by that enclave. Information that is paged-out from the EPC into regular

[2]Side-channel attacks that compromise confidentiality of SGX enclaves have been identified [71, 39]; we assume such attacks cannot undermine the confidentiality of tagged values handled by authorized enclaves.

DRAM is encrypted under a hardware-derived key.

Data sealing. SGX enclaves are uniquely identified by an SGX Enclave Control Structure, which includes a *measurement*—a 256-bit digest of a cryptographic log recording the build process for the enclave. This measurement is used by the key generation instruction (along with secrets embedded in the SGX chip) to produce hardware-derived sealing keys. Sealing keys for an enclave depend on both the measurement of the enclave and the hardware keys of the chip; sealed data can only be decrypted by the enclave that originally sealed it. Data sealing can provide confidentiality and integrity for audit logs and for tagged values that will be temporarily stored or handled outside the enclave.

Local Attestation. Enclave measurements are also used for local attestation, which allows one enclave to authenticate the program that is running in another enclave. Local attestation (between enclaves) uses a hardware-signed (HMAC'd) copy of the enclave measurement—the *report*—combined with a Diffie-Hellman key exchange protocol to prove the identity of the program in one enclave to the second enclave. Local attestation is used for local program authentication.

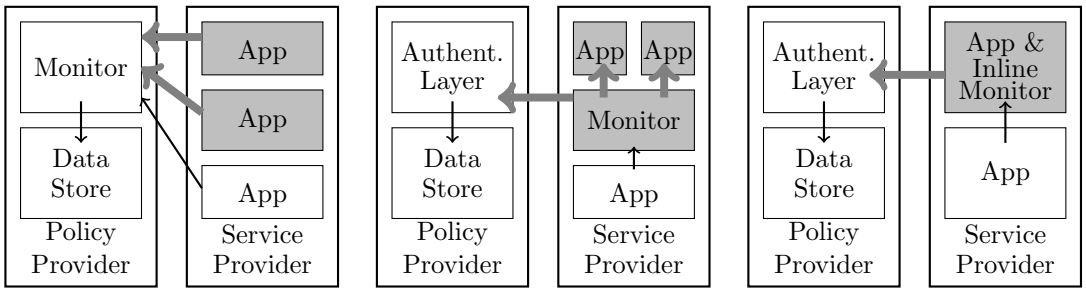
Remote Attestation. SGX implements remote attestation using local attestation together with a pair of dedicated, Intel-authored enclaves: a *provisioning enclave* and a *quoting enclave*. The provisioning enclave requests an attestation key from Intel and stores it sealed under a key that can only be derived by Intel-authored enclaves. The quoting enclave retrieves the attestation key, verifies the measurement using local attestation, and signs the measurement together with an optional message; the resulting signed measurement-message pair, called a *quote*,

can be verified using Intel's Attestation Service.

Remote Authentication. Because communications between an application enclave and the quoting enclave are mediated by an untrusted (i.e., non-enclave) application, quotes can be intercepted and replayed by any program. To mitigate this threat, our remote attestation protocol requires the application enclave to fetch an application secret $\langle s_1, s_2 \rangle$—where $s_2 = H(s_1; k_{RS})$ and k_{RS} is a secret key unique to the remote server RS—from RS across a secured communication channel; this secret is used to prove to RS that a quote was generated by the principal (an application enclave) at the end of the secured channel. To do so, an application enclave sets s_2 as the message used during measurement generation and then requests a quote, resulting in a quote $q(s_2)$. To perform remote authentication, the application enclave sends the pair $\langle s_1, q(s_2) \rangle$ to the remote server. The server authenticates the secret by checking that the quote message s_2 is equal to $H(s_1; k_{RS})$, authenticates the quote with Intel's attestation service, and then uses the authenticated quote to make an authorization decision.

4.2 Enforcement by Source-based Monitoring

The first step in designing an enforcement architecture is to decide which principal will be trusted to perform the monitoring. Principals are either policy providers or service providers; a monitor can be run at either. Since policy providers are trusted, it is natural to have policy providers run the monitors. In this source-based monitoring architecture, SGX can be used to determine which applications (running remotely at a service provider) are authorized to use a given value. Assuming that sensitive values can be processed only by a standard set of data analytics



(a) Source-based Monitoring (b) Delegated Monitoring (c) Inline Monitoring

Figure 4.1: An overview of the different architecture designs. The direction of each arrow indicates which principal instigates communication between two components of the system. SGX enclaves are shown in gray; wide gray arrows indicate that a program has authenticated using an SGX report (local attestation) or quote (remote attestation).

functions, a source-based monitor can distinguish between authorized and unauthorized applications and, therefore, can enforce use-based privacy in the presence of malicious service providers. Moreover, with all policy enforcement performed at the policy provider, application developers do not need to handle policies or explicitly interact with policy mechanisms, and policy enforcement is compatible with legacy applications.

4.2.1 Designing a Source-based Monitor

Applications run by a service provider issue requests $\langle r, x, c \rangle$ to a policy provider, where r is the type of request (e.g., `GET values`), x is a reference to the requested data (required for requests that retrieve values), and c is a set of credentials. Traditional authentication tokens—e.g., OAuth tokens or signed statements—can attest to the invoker type I and the purpose type P; we use SGX quotes as credentials for the executable type E, as described in Section 4.1. Upon receiving a request, the monitor validates the request, it retrieves the requested values (and their policy

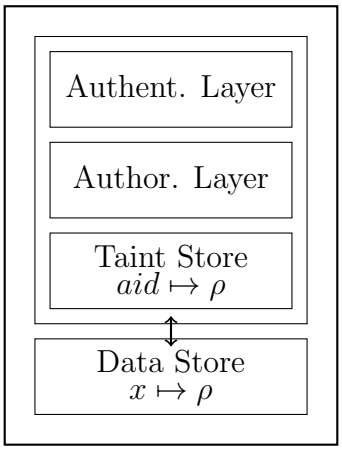


Figure 4.2: Detailed design for a policy provider that implements source-based monitoring.

tags) from the data store, and then constructs a policy-compliant response. This architecture is depicted in Figure 4.1a, and details (discussed below) are shown in Figure 4.2.

To construct a policy-compliant response to a request for data, the monitor invokes an *authentication layer* to authenticate the *request credentials* and determine the use type $(\mathsf{I}, \mathsf{E}, \mathsf{P})$ for the application that issued the request. We consider two possible approaches. In a prevention-based monitor, the authentication layer compares the authenticated credentials to a whitelist of known credentials in order to determine the use type. This results in a monitor that enforces policy compliance against malicious adversaries. In a detection-based monitor, the authentication layer creates a log entry—including an identifier for the service provider, the authenticated credentials, and the claimed use type—and then accepts the claimed use type. Because a service provider could misstate the use type, a detection-based monitor does not enforce policy compliance against malicious adversaries. Observe, however, that the audit log ensures that incorrect use types can be detected after the fact, so a detection-based monitor is sufficient to guarantee policy compliance

in the presence of accountable adversaries.

After determining the use type, the monitor retrieves the requested values (and their policy tags) from the data store. It then invokes an *authorization layer*, which compares the use type to the use-based privacy policy defined by the policy tags and constructs a policy-compliant response. The details of how this response is constructed are implementation-specific and are discussed in Section 4.2.2.

Since we express use-based privacy as Avenance policies and since Avenance policies are reactive, the monitor is also responsible for computing derived policies and associating them with derived values. To do so, the monitor maintains a *taint store* that maps applications to the Avenance policy(s) of the values that application has received. Each time the monitor sends values to an application, it adds the corresponding policies to the taint store entry for that application. When the monitor receives a new value x from an application, it first invokes the authentication layer to authenticate the request credentials and determine the use type $(\mathsf{I}, \mathsf{P}, \mathsf{E})$ and the application identifier *aid*. It then looks up the policy(s) ρ associated with *aid* in the taint store, invokes the transition triggered by the executable type $\mathsf{E}(aid)$ to produce a derived policy ρ', constructs a tagged value $\langle x, \rho' \rangle$, and stores the new tagged value in the data store.

4.2.2 Implementation of Source-based Monitoring

A policy provider stores tagged values and distributes those values to service providers; an implementation of a policy provider is described in Chapter 5. We extended that policy provider to add source-based policy enforcement. The extended policy provider is implemented in 7634 lines of Java on top of an existing

Ohmage [60, 67] data store.

Policy Enforcement. To ensure policy compliance, our extended policy provider only accepts requests received over a TLS connection and accompanied by request credentials. Credentials might include an OAuth token, a purpose label, and/or an SGX quote and nonce n. OAuth credentials are authenticated and then used to lookup the service provider identity *spid*—a unique identifier associated with an OAuth client secret. Purpose labels are not authenticated; they are instead interpreted as credentials of the form "U says P" for the user U defined by the OAuth token and some purpose type P. SGX quotes are authenticated with the Intel Attestation Service and then cached; the quote is also used to define the application id *aid*. Finally, the policy provider determines the executable type: if configured for prevention-based enforcement it compares the quote to a whitelist of trusted enclaves, if configured for detection-based enforcement it accepts the claimed enclave type after logging the request. The enforcement mode is set at runtime using the command line argument `--ENF.MODE`. The policy provider then performs monitoring as described in Section 4.2.1.

4.2.3 Evaluation of Source-based Monitoring

We deployed our policy provider with source-based monitoring on an Amazon EC2 T2.small instance with an Intel Xeon E5-2676 2.4 GHz CPU and 2GB of memory running Ubuntu 14.04 LTS (kernel version 3.13.0).

To evaluate performance, we measured latency and throughput of the policy provider responding to a `GET datapoints/scope` request for 500 datapoints. We

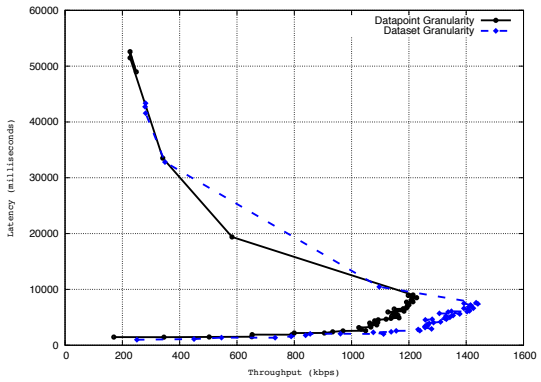


Figure 4.3: Latency and throughput of a policy provider with source-based monitoring as a function of the number of concurrent requests, ranging from 1 to 50 concurrent requests. The solid black line shows performance when the policy provider enforces datapoint-granularity policies and the dashed blue line shows performance for dataset granularity.

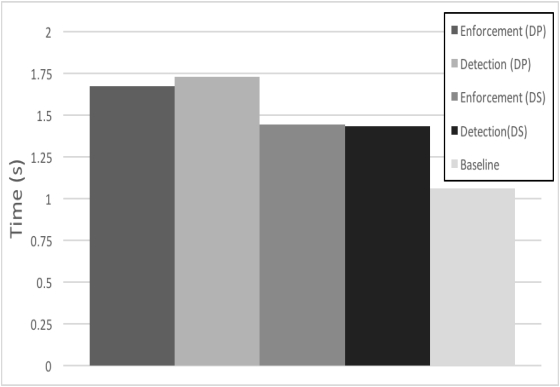


Figure 4.4: Performance of the PMSys averaging function with source-based monitoring.

tested the performance as the policy provider handled between 1 and 50 concurrent requests. As shown in Figure 4.3, implementation choices—for example, whether policies were associated with values at the granularity of individual datapoints or for the full dataset—did impact the performance. But all implementations overloaded at a relatively low load (less than 50 simultaneous requests), after which throughput collapsed and latency drastically increased.

We also measured the performance of source-based monitoring for a common

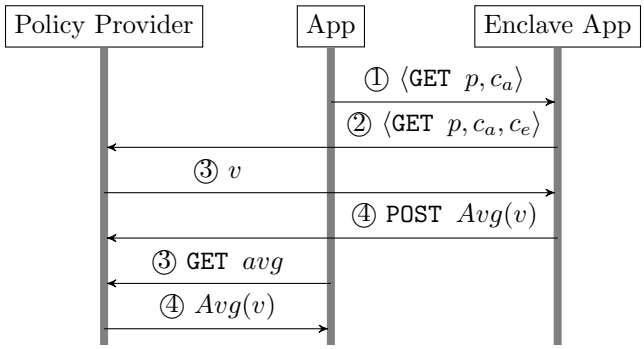


Figure 4.5: Protocol for the PMSys averaging function in a source-based monitoring architecture.

use case [5]: user preferences, legal regulations, and/or contractual terms restrict uses for raw data but allow derived values (e.g., anonymized values, encrypted values, or aggregated values) to be used more liberally. One such policy is depicted in Example 2.9. A compliant service provider might first request the raw data, generate the derived values, and then use the derived values.

To evaluate performance for this use case, we ported one such application, called PMSys [56], to run on the source-based monitoring architecture. PMSys is a mobile and web-based application developed jointly at Simula Research Laboratory and UIT The Arctic University of Norway that performs physiological evaluation and training-load personalization for soccer players. PMSys collects data about player mood, sleep patterns, physical fitness, and injuries and displays aggregate statistics (including average) to authorized coaches. These data are subject to data-use restrictions derived both from the data-use contract signed by the players (who all are members of elite clubs and national teams in Norway, Sweden, and Denmark) and relevant national and EU privacy laws that restrict data sharing.

We measured the end-to-end latency of the PMSys averaging function on syn-

thetic data matching the PMSys data collected for one month.[3] To eliminate network bottlenecks, we ran our policy provider on a dedicated Amazon EC2 R4.large instance with an Intel Xeon dual-core E5-2686 2.3 GHz CPU and 15GB of memory running Ubuntu 14.04 LTS (kernel version 3.13.0). We deployed the application on an OptiPlex 5040 with an SGX-enabled Intel Core i5-6500 3.20 GHz CPU and 16GB of memory running Ubuntu 14.04 LTS (kernel version 4.4.0). As shown in Figure 4.4, this averaging function experiences 35–62% overhead compared to a baseline averaging function with no policy enforcement. However, poor performance is unsurprising given the number of round-trips required; as shown in Figure 4.5, this averaging function requires three round trips to the server in order to enable the source-based monitor to mediate access to the derived (average) value.

4.3 Enforcement by Delegated Monitoring

To mitigate the throughput bottleneck imposed by the monitor in a source-based monitoring architecture, we turned to an alternative design. In a *delegated monitoring* architecture, service providers run the monitors in dedicated SGX enclaves, which enables each monitor to authenticate itself to the policy provider. We assume that there will only be a small number of implementations of delegated monitors, so a policy provider can whitelist the credentials for delegated monitors to ensure that tagged values are shared only with valid instances of a delegated monitor. SGX is used here also to determine locally which applications run by the service provider are authorized to use values, and it is used to provide confidentiality and

[3]Using actual data from the production system would have been incompatible with the existing terms of service and Norwegian data protection laws.

integrity for tagged values handled by a delegated monitor. As before, we assume that sensitive values can only be processed by a standard set of data analytics functions, so a delegated monitor can distinguish between authorized and unauthorized applications and, therefore, can enforce use-based privacy in the presence of malicious service providers.

A delegated monitoring architecture requires each service provider to run a monitor. Because each monitor is responsible for mediating the requests from just one service provider, the delegated monitoring architecture eliminates the performance bottleneck incurred by a source-based monitor. This architecture also offers an opportunity to mitigate the second performance drawback of source-based monitoring: the number of round-trips required for a typical application. In the source-based architecture, it is necessary to send all derived values to the policy provider, because the monitor (run by the policy provider) mediated all requests, including requests for derived values. Because a delegated monitor is run locally by a service provider, those round trips are no longer necessary. Instead policies pertaining to derived values can be determined by the local monitor, and derived tagged values can be cached locally using SGX sealing. This design improves performance at the cost of introducing a burden on application developers, who must now handle tagged values and must modify any legacy applications.

4.3.1 Designing a Delegated Monitor

Delegated monitors run by a service provider act as a proxy for untrusted applications: they issue requests to a policy provider and they mediate messages to and from enclave applications. This architecture is depicted in Figure 4.1b, and an example sequence of interactions is depicted in Figure 4.6. The design of

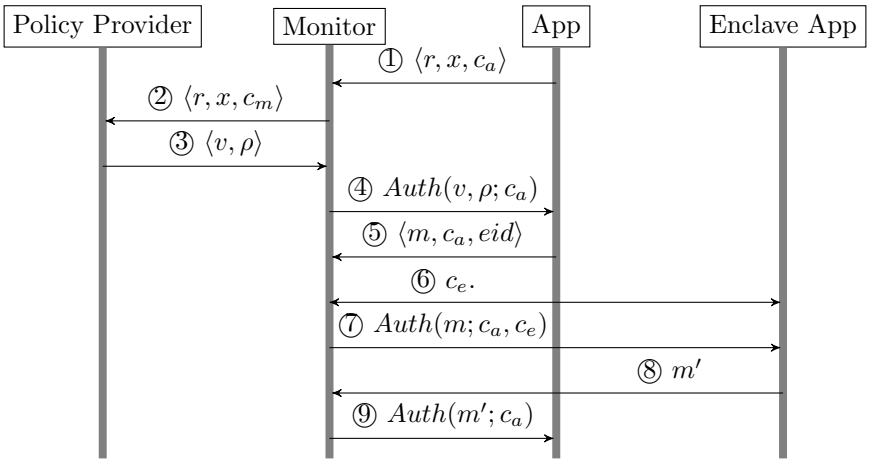


Figure 4.6: Example interactions with a delegated monitor.

the delegated monitor is the same as the source-based monitor design depicted in Figure 4.2.

Delegated monitors accept requests $\langle r, x, c_a \rangle$ from untrusted applications, where r is the type of request (e.g., `GET values`), x is a reference to the requested data (required for requests that retrieve values), c_a is a set of invoker credentials (e.g., message ① in Figure 4.6). A monitor then replaces the credentials c_a with a set of monitor credentials c_m and issues the modified request ② to a policy provider in the form $\langle r, x, c_m \rangle$. Upon receiving the request, the policy provider authentication layer checks the monitor credentials and then issues a response ③. After a monitor receives a response from a policy provider, it mediates the response to enforce policy compliance. If the response contains no tagged values (e.g., an acknowledgment), then it forwards the response to the untrusted application. If the response contains tagged values $\langle v, \rho \rangle$, then the monitor invokes an authorization layer, which compares the use type of the untrusted application $(\mathsf{I}, \mathsf{P}, null)$[4]—determined by the internal authentication layer from the application credentials

[4]Note that the use type cannot define an executable type because untrusted applications do not run inside SGX enclaves and therefore cannot produce the necessary credential—a quote—for an executable type E.

c_a—to the use-based privacy policy defined by the policy tags and constructs a policy-compliant response ④ $Auth(v, \rho; c_a)$. The details of how this response is constructed depend on the granularity of the policy tags returned by the policy provider, but the monitor forwards authorized values to the untrusted application in plaintext and encrypts all other values using an SGX sealing key.[5]

Delegated monitors also mediate messages between untrusted applications and trusted applications. Communication is always initiated by an untrusted application, which sends a message $\langle m, c_a, eid \rangle$ where m is either a set of tagged values or sealed tagged values, c_a is a set of invoker credentials, and eid is an enclave application (e.g., message ⑤ in Figure 4.6). The monitor authenticates the invoker credentials to determine the invoker type I and purpose P, and then authenticates the enclave application eid ⑥ and determines the executable type E. It then invokes the authorization layer, which compares the use type $(\mathsf{I}, \mathsf{P}, \mathsf{E})$ to the use-based privacy policy defined by the (decrypted, if necessary) policy tags, constructs a policy compliant message ⑦ $Auth(m; c_a, c_e)$ (using SGX sealing, if necessary), and forwards the resulting message to enclave eid. It also updates the taint store entry for eid to include the policies for any tagged values sent to eid in plaintext. When the monitor receives a response—a set of values ⑧ m'—it looks up the policy ρ associated with eid in the taint store, invokes the transition triggered by the executable type E to produce a derived policy ρ', and constructs a new set of tagged values from m' and ρ'. Finally, it invokes the decision engine to determine whether ρ' authorizes the untrusted application $(\mathsf{I}, \mathsf{P}, null)$, constructs a policy compliant response ⑨ $Auth(m'; c_a)$ (using SGX sealing, if necessary), and

[5]This design eliminates unnecessary round-trips to the policy provider by caching encrypted copies of tagged values with any application that is not authorized to use those values. This caching might violate a use-based privacy policy unless we interpret policies as allowing encrypted copies of tagged values to be used by any principal in any way. We consider such an interpretation consistent with existing user preferences and legal requirements.

forwards that response to the untrusted application.

4.3.2 Implementation of Delegated Monitoring

Policy Provider. We modified our policy provider to work in concert with delegated monitoring. It retains the same data store and policy association API as in the source-based monitoring architecture, and it, too, can be configured to construct tagged values at either datapoint granularity or dataset granularity.

Instead of mediating requests to enforce policy compliance, the modified policy provider uses an authentication layer to accept requests only over TLS from delegated monitors. The policy provider authentication layer authenticates credentials $\langle s_1, q, e \rangle$ as described in Section 4.1 and determines the use type E using the same authentication mechanism—either prevention-based or detection-based—as the source-based monitor in Section 4.2. If the requester successfully authenticates as a delegated monitor—denoted by the special use-type $\mathsf{E} = \texttt{policyrm}$—the policy provider returns the requested tagged values.

Delegated Monitor. We implemented a delegated monitor in 1149 lines of C/C++ that runs as a dedicated SGX enclave. On initialization, the monitor establishes its credentials $\langle s_1, q, e \rangle$ for use in remote program authentication, as described in Section 4.1: it retrieves an application secret (s_1, s_2) from the policy provider, generates a quote q with message s_2, and defines $e =$ `policyrm`. All subsequent requests to the policy provider are sent over TLS using a version of the `mbedtls` client ported to run inside an SGX enclave [72]; these request include the monitor credentials as a message header.

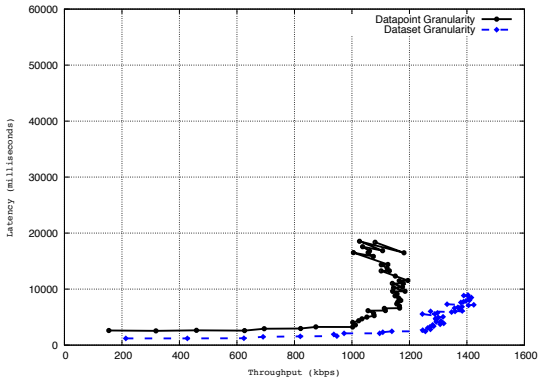


Figure 4.7: Latency and throughput of a policy provider with client-side monitoring (delegated monitoring or inline monitoring).

Policy Granularity. Like the source-based monitoring implementation, our implementation of delegated monitoring supports policy tags at two different granularities: individual datapoints and datasets.

Policy Enforcement. For efficiency, our delegated monitor exclusively implements prevention-based monitoring; it determines use types $(\mathsf{I}, \mathsf{P}, \mathsf{E})$ by comparing invoker and enclave credentials to a whitelist of known types.

4.3.3 Evaluation of Delegated Monitoring

We deployed our policy provider with delegated monitoring on the same Amazon EC2 instances and the same local client that we used to evaluate source-based monitoring.

We evaluated the performance of the policy provider in the delegated monitoring architecture by reproducing the latency and throughput experiment we ran for the source-based monitoring architecture. The simplified authentication layer run

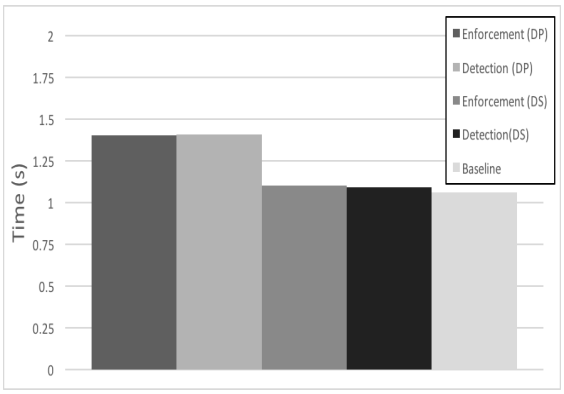
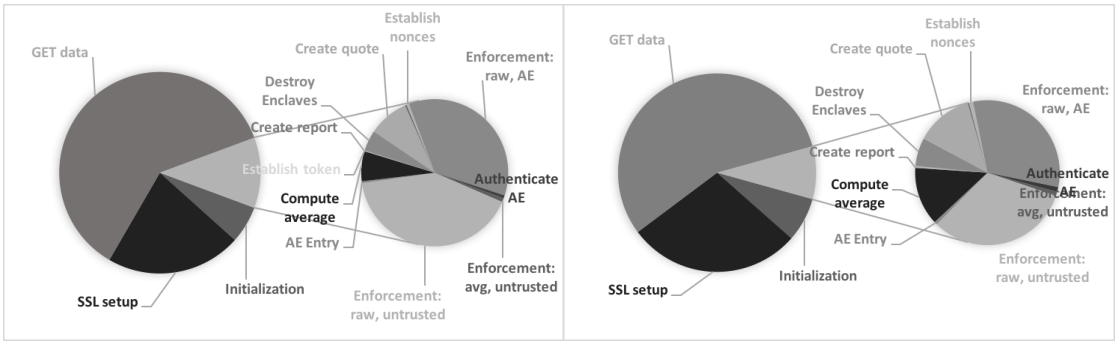


Figure 4.8: Performance of the PMSys averaging function with delegated monitoring.



(a) Datapoint granularity (b) Dataset granualarity

Figure 4.9: Breakdown of the latency of the PMSys averaging function with delegated monitoring.

by the policy provider eliminates the throughput bottleneck incurred by a source-based monitor; this improved performance is evident for both datapoint granularity and dataset granularity (Figure 4.7). Observe that implementing policy association at the granularity of datasets results in a moderate increase in throughput and a significant decrease in latency, as compared to the datapoint-granularity implementation.

To evaluate the performance of the delegated monitor for the common aggregate-then-use case, we ported the PMSys application—which requests values, com-

putes the average in an SGX enclave, and then uses the average in an untrusted application—to run in the delegated monitoring architecture. The reduced number of round trips significantly improves the performance of the averaging function, as compared to the source-based monitoring architecture (Figure 4.8); there is a 3% overhead for dataset-granularity and the overhead is cut in half for datapoint-granularity enforcement. Significant components of the remaining overhead are due to the cost of sealing cached values (Figure 4.9). The majority of the latency is due to enclave initialization, SSL negotiation, and fetching the raw data; these costs are fixed. The majority of the remaining latency can be attributed to the cost of enforcing policy compliance when caching raw data with the untrusted application (which requires sealing the data) and when transferring cached, raw data to the averaging enclave (which requires unsealing the data). This cost is likely to increase for applications that handle more data (much of the difference in latency between datapoint and dataset mode is due to the increase space required to store policies at the granularity of individual datapoints).

4.4 Enforcement by Inline Monitoring

To eliminate the latency overhead imposed by sealing cached tagged values, we propose yet a third design. In an *inline monitoring architecture*, the service provider performs monitoring inline with a monitored application, like the Tir system described in Chapter 3.2,. The inline architecture enables applications to process tagged values within a single enclave, eliminating the need to seal cached values. But, as with Tir, application developers must instrument their code with monitor calls, which is a burden.

To ensure policy compliance, a policy provider must send tagged values only to correctly-inlined applications running inside SGX enclaves. With many correctly-inlined applications, a policy provider cannot be expected to maintain a database identifying all. Prevention-based monitoring—in which the policy provider authentication layer maintains a whitelist of authorized enclaves—is therefore infeasible.[6] Instead, we focus on detection-based monitoring, and we design and implement an inline monitor that will ensure policy compliance by accountable service providers.

4.4.1 Implementation of Inline Monitoring

We modified our Tir library described in Chapter 3 to run inside SGX enclaves. When the monitor implemented by the library is configured with logging, it generates a secure audit log that contains a record for each invocation of a monitor call that affects the state of the monitor—store_policy, delete_policy, trans, and change_use—and each time enforcement occurs—each invocation of use. A record contains the monitor call, the arguments to the monitor call, and a record id (a counter that is increased with each record). Each entry is encrypted using SGX sealing and then written to a logfile stored in the local file system. Log records cannot be modified because SGX sealing ensures integrity, and the counter ensures that log records cannot be deleted. Note that an auditor uses the application to retrieve (and unseal) the audit log—the correctness of this function is ensured because the policy provider logs the application quote—and the retrieval function includes the current counter value, so truncations of the audit log can also be

[6]The preceding architectures do not have this constraint. In either a source-based monitoring architecture or a delegated monitoring architecture, all service providers might use a common set of data analysis enclave applications to manipulate tagged values; in a delegated monitoring architecture, the policy provider must also authenticate the delegated monitor, but each service provider runs an instance of the same (or one of a small number of) monitor enclave, so prevention-based enforcement is a feasible option.

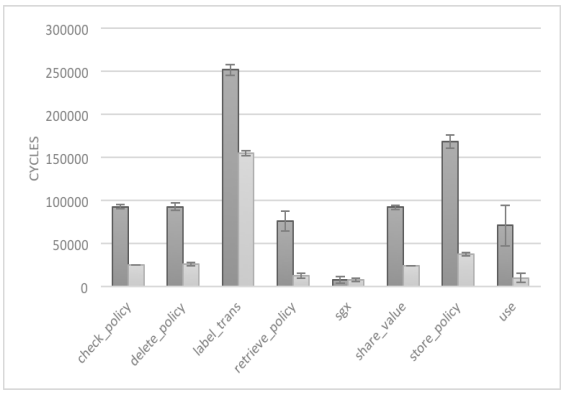


Figure 4.10: Performance of inline monitoring library calls.

detected.

4.4.2 Evaluation of Inline Monitoring

The inlined applications are compatible with the policy provider implemented for delegated monitoring, so the policy provider exhibits the same performance shown in Figure 4.7.

To evaluate the performance of the inline monitor, we ran a series of microbenchmarks that evaluate the costs of various library calls. These results are shown in Figure 4.10. We find that the detection-based implementation has higher latency due to the additional cost of encrypting log entries with SGX sealing, exiting the enclave, and writing the log entries to the logfile.

To evaluate the performance of the inline monitor for the common aggregate-then-use case, we ported the PMSys application—which requests values, computes the average in an SGX enclave, and then uses the average in an untrusted application—to run in the inline monitoring architecture in prevention mode. As shown in Figure 4.11, inline monitoring is within the error margin of the baseline

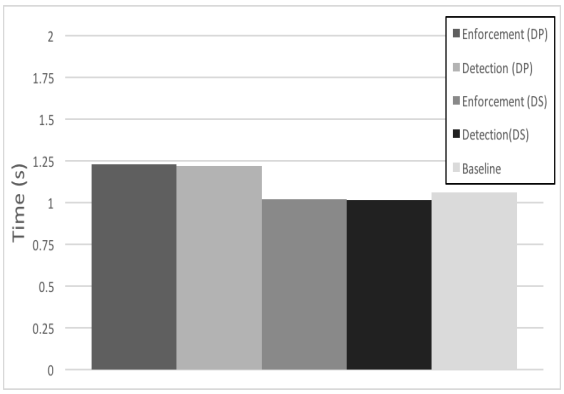


Figure 4.11: Performance of the PMSys averaging function with inline monitoring.

Architecture	Security	Performance	Deployability
Source-based	malicious adversaries (✓)	poor (−)	No programmer burden (✓)
Delegated	malicious adversaries (✓)	moderate (∼)	some policy handling (∼)
Inline	accountable adversaries (∼)	good (✓)	significant annotations (−)

Figure 4.12: Tradeoffs between different monitoring architectures. ✓indicates goals that are fully met , ∼ indicates goals that are partially met, − indicates the architecture failed goals.

system for dataset-granularity enforcement, and it offers significantly improved performance (14% overhead) with datapoint granularity However, this performance comes at the cost of increase the burden on application developers and an attenuated threat model.

4.5 Comparing Approaches

In this chapter, we discuss the feasibility of using Intel SGX as a root of trust to enforce such policies in the presence of an active adversary. The natural, source-based monitoring architecture enables policy enforcement against malicious adversaries with minimal effort for application developers, but it brings significant performance overhead. So we explored two alternative architectures—delegated monitoring and

inline monitoring—that offer improved performance and that demonstrate a trade-off between deployability, security, and performance. A delegated monitoring architecture provides the best performance against malicious adversaries, but an inline monitoring architecture provides performance improvements—particularly for applications that handle more data or require finer-grained policies—in an attenuated threat model. Given the trade-offs between deployability, performance, and security (summarized in Figure 4.12), we believe that the appropriate architecture will depend on the type of application. However, we view our results as positive evidence of the feasibility of enforcing high-level use authorizations and privacy policies in a decentralized, adversarial ecosystem.

CHAPTER 5

POLICY MANAGEMENT FOR USE-BASED PRIVACY

In the preceding chapters, we assumed that use-based privacy policies—perhaps expressed as Avenance policies—would be associated with values to create tagged values. Systems would operate on tagged values, ensuring that a policy would be available whenever a sensitive value was used, thereby enabling enforcement. In this chapter, we investigate the problem of how to associate policies with values.

A naïve approach might require each user to establish an account with each service provider, report values to each service provider, and associate each copy of each value with the appropriate policy. This approach, however, imposes an unreasonable burden on users. The burden of policy association might be alleviated by requiring service providers perform policy association, but the resulting approach would still burden users with the need to establish accounts with every service provider while simultaneously requiring service providers to duplicate the work of policy association. A better approach would delegate account management and policy association to one, or a few, trusted principals who then mediate access to the resulting tagged values.

This last approach is analogous to an *identity-management system*. Identity management systems were developed as a reaction to problematic practices concerning online identities: each service provider maintained a set of user identities, so users had many identities (at least one for each service provider with which they interacted), and this multitude of identities became a management burden as well as creating many potential points of failure. Identity-management systems enhance security and convenience by introducing a new class of principals called *identity providers*.

In this chapter, we survey existing identity management systems, and we draw on the lessons learned from identity management to develop a policy provider for storing values and associating use-based policies with user data.

5.1 Identity Management Systems

In an identity management system, an identity provider is trusted to perform certain functions.

(1) It is responsible for authenticating users—i.e., determining whether a particular user is associated with a particular identity—and issuing authentication assertions in support of these authenticated users. The manner in which an identity provider authenticates users could affect whether service providers accept authentication assertions issued by that identity provider.

(2) It stores *identities*—i.e., collections of attributes—for users and is responsible for managing these identities. Details vary across systems but, generally, an identity provider would have provisions for creating, updating, releasing, and deleting attributes and identities.

(3) It provides evidence or *authentication assertions* that service providers can use to make authorization decisions.

Service providers depend on receiving authentication assertions that contain current attributes, so an identity provider is not only be responsible for validating attributes initially—that is, verifying their value in connection to a real-world identity—but might also maintain currency of those attributes. The methods used for validating attributes (including whether to trust the user or to perform independent validation of claimed attributes) and for eliminating or updating stale

attributes often determine whether a service provider will accept authentication assertions issued by some identity provider.[1]

Existing identity management systems[2] can be divided into three classes:

- *Single Sign-on*: Systems designed to issue authentication assertions to multiple service providers after a single user authentication. Examples include Passport[3] [44], OpenID[4] [50], Shibboleth [64], and Facebook Single Sign-On [25].
- *Federated Identity*: Systems designed to manage multiple distinct identities for a single user and to issue authentication assertions on the basis of any of these identities. Examples include Project Liberty[5] [59], Higgins [30], PRIME [58], CardSpace [16], and Client-Side Federation [13].
- *Anonymous Credentials*: Systems designed to provide authentication assertions that do not reveal the user's identity to a service provider. Examples include Idemix [11, 12], U-Prove [51], and P-IMS [34, 65].

Identity management systems can also be characterized by some key design choices; these design choices revolve around each system's solutions the problems of attribute storage and attribute confidentiality.

[1] In some systems, attribute validation, management, and/or storage are delegated to a separate party called the *attribute provider*.

[2] Detailed overviews of some well known identity management systems appear in the Appendix D.

[3] Microsoft's Passport is no longer supported; this system is being included for historical reasons.

[4] Although our primary focus is on systems and implementations, we discuss the OpenID specification because it has no representative implementation.

[5] Project Liberty was merged into SAML 2.0 (the specification implemented by Shibboleth and various enterprise identity solutions) and is no longer actively maintained. It is included for historical reasons, because there are no existing representative implementations.

5.1.1 Attribute Storage

In identity management systems, identities are stored and maintained by identity providers. Existing systems have developed several approaches regarding the nature and distribution of identity providers; these approaches differ in their support for *unlinkability*—a property that concerns permanently or temporarily hiding correlations between identities.

Centralized. A *centralized identity management system* is one in which a single, dedicated identity provider manages all identities. In systems that adopt this design, a user must trust the identity provider, and all of that user's attributes are disclosed to this identity provider. Note, when a user chooses to create multiple distinct identities (e.g., a personal identity and a work identity), patterns in attribute value or use could allow the identity provider to link multiple identities to the same individual, because all identities are stored by the same identity provider. Since information that links user identities and actions performed under those identities is economically valuable, identity providers have an incentive to favor a centralized identity management system. Passport and Facebook Single Sign-On are examples of centralized systems.

Federated. A *federated identity management system* represents an intermediate design point in which a user chooses which identity providers to trust with links between certain identities associated with that user. Project Liberty supports this. There, identity providers that have established business relations form *circles of trust*. Within a circle of trust, a user can opt to *federate* two identities, in which case the identity providers exchange information and the identities are linked.

	Passport	Liberty	Idemix	Shibboleth	Higgins	PRIME	OpenID	CardSpace	U-Prove	CS-Federation	P-IMS	Facebook SSO
Centralized	✓	×	×	×	×	×	×	×	×	×	×	✓
Federated	×	✓	×	×	×	×	×	×	×	✓	×	×
Decentralized	×	×	✓	✓	✓	✓	✓	✓	✓	×	✓	×

Figure 5.1: Existing approaches to attribute storage

Client-Side Federation is also a federated identity management system, but it is designed to ensure that local handles are only known to the user (not the identity provider) thereby guaranteeing privacy against identity providers that collude with service providers.

Decentralized. A *decentralized identity management system* includes multiple, distinct identity providers that each function separately, possibly using different protocols, and may not even be aware of each other. A user can create one or more identities with any identity provider in the system. This architecture allows users to choose which identity providers to trust with which attributes, and it also allows a user to distribute sensitive attributes across distinct identity providers, thereby ensuring unlinkability of distinct identities. The approach has been widely adopted by existing identity management systems (e.g., Idemix, Shibboleth, Higgins, PRIME, OpenID, CardSpace, U-Prove, and P-IMS).

5.1.2 Attribute Confidentiality

Principals can learn user attributes through three channels: intentional channels (P receives the attributes directly from U or from some identity provider IDP acting on behalf U), attribute forwarding (P acquires the attributes from another party P'), and attribute inference (P deduces the attributes from other known or observed information).

Intentional channels are instigated by users. Design choices here focus on how users control what attributes to release, when, and to which principals. These design choices can be applied both to direct release of attributes by a user and to dissemination of attributes by an identity provider acting on behalf of a user.

Instance-based Attribute Release. Under this approach, the user explicitly specifies whether information may be released to a principal that requests it. Instance-based attribute release is extremely flexible, since users can make decisions based on any factors available at the time the request occurs. However, this approach can be inconvenient, since it requires the user to make many decisions and since determining which service providers to trust can be difficult. Nonetheless, many existing identity management systems today provide instance-based attribute release to control the dissemination of attributes to service providers. CardSpace and Higgins service providers send the user a policy (described using HTML or WS-SecurityPolicy) specifying required attributes and authentication assertion types. The user must then (interactively) decide whether to share the requested attributes with the service provider and, if so, which available identity to use. Facebook Single Sign-on presents the user with permissions requested by the service provider, which the user must choose to accept or deny. Mechanisms

for controlling attribute release are beyond the scope of the OpenID specification, but the OpenID Attribute Exchange protocol specification does permit responses that indicate an attribute is not available (using `attribute.count=0`). The Google OpenID identity provider leverages this capability and gives users an option to permit or deny the release of each attribute required by the service provider (with an option to remember selected permissions for future interactions with that service provider).

Policy-based Attribute Release. With this approach, a user defines a *policy*, and an attribute is released to a service provider if and only if the specified policy is satisfied for that attribute. The policy can involve attribute type, attribute value, party identity, or properties of the party when defining the conditions for attribute release. In a typical implementation, the identity provider (or an active client acting for the user) will automatically determine which parties should receive specific attributes on the basis of the user's policy. Although policy-based attribute release is typically less expressive than instance-based (depending on the language in which policies are written), policy-based release simplifies the user experience by automating the dissemination of attributes. Policy-based attribute release is employed by Passport, Shibboleth, and PRIME. Passport employs a very restrictive language for expressing user policies. A user labels each attribute as `public` or `private`. Attributes are then automatically released to other parties based on these labels: attributes labeled `public` are sent to any service provider with which the user chooses to interact. In Shibboleth, attribute release is controlled by an *attribute filter*, which is a collection of policy rules. Each *policy rule* consists of a single *requirement rule*—it determines whether the policy rule is binding for a particular authentication request—and zero or more *attribute rules*—these

specify which attributes are governed by the policy rule and whether or not those attributes will be released. Each policy rule is expressed using a flexible policy language and can depend on each of the parties involved as well as attribute type and value. The PRIME system makes extensive use of policies that govern data access, data release, and data handling. A user and a service provider both define policies. Starting from these, the system automatically negotiates conditions of data release; this negotiation can, but does not necessarily, include active user input. A successful negotiation finds a solution that satisfies both parties' policies; such a solution is required before any attribute will be disclosed. PRIME software, running locally at the service provider, is designed to automatically enforce any conditions that are agreed during the negotiation.

Attribute forwarding occurs when a party P acting on its own initiative discloses an attribute to another party P'. An example is when a service provider that sells goods forwards a user's address to a service provider that arranges delivery. Attribute forwarding is invisible to the user, but some identity management systems introduce mechanisms that allow users to prevent or control such disclosures, nevertheless.

Deniable Attributes. *Deniable attributes* cannot be validated without cooperation from an identity provider or user. This embodies the philosophy that the significant concern about attribute release is whether parties can prove to others that a user satisfies some particular attribute—not whether those parties can merely learn or claim the user satisfies those attributes. The approach is widely adopted in existing identity management systems. Systems that employ interactive authentication implement deniable attributes by encrypting assertions with a

shared secret key (Passport and OpenID) or by sending service providers a reference that will be validated interactively only with explicit user permission (Project Liberty and Shibboleth). Systems that employ credential-based authentication can implement deniable attributes by requiring the presenting party to know a particular secret in order for the credential to be validated (Idemix, U-Prove, and P-IMS).

Policy Tags. Another approach to control attribute forwarding involves the use of policy tags. PRIME supports policy tags to implement control over data use (including attribute forwarding); a policy negotiation phase occurs prior to attribute release, and all released attributes are tagged with policies that are mutually agreed during that phase. Policy tags can specify notification requirements, deletion requirements, and limitations on forwarding or other uses of attributes. PRIME relies on a combination of legal accountability and secure hardware as the basis for users to trust that policy tags are enforced; legal accountability ensures service providers use correctly-installed trusted hardware modules, the trusted hardware generates cryptographic assertions that tagged attributes are accessed only by PRIME software, and legal accountability can be invoked if correct assertions are not received.

Attribute inference occurs when a party P can deduce the value of an attribute from other information known to P. Attribute inference, like attribute forwarding, is not controlled by users, so design choices again concern whether and how to control or prevent principals from learning (or using) inferred attributes.

Attributes can be inferred from user behavior, including websites visited and

		Passport	Liberty	Idemix	Shibboleth	Higgins	PRIME	OpenID	CardSpace	U-Prove	CS-Federation	P-IMS	Facebook SSO
Intentional	Instance	×	?	✓	✓	✓	×	?	✓	✓	✓	✓	✓
	Policy	✓	?	×	✓	×	✓	?	×	×	×	×	×
Forwarding	Deniable	✓	✓	✓	✓	?	✓	✓	?	✓	×	×	×
	Policy Tags	×	×	×	×	?	✓	×	?	×	×	×	×
Inference Mitigation		×	×	×	×	?	✓	×	?	×	×	×	×

Figure 5.2: Existing approaches to attribute confidentiality

items viewed or purchased, or from other attributes.[6] One standard defense against attribute inference is to minimize the amount of information that P learns—this can be done by limiting the types of user behavior that can be detected or observed by P and by limiting the attributes that P can learn over other channels. If a party has prior access to information about particular user action or attribute, then it is impossible to prevent that party from learning whatever attributes can be inferred from that information alone. However, in many cases, accurate attribute inference requires linking many different pieces of information and even performing statistical analysis. A second standard defense is, therefore, to prevent linking between attributes disclosed in different interactions.

5.2 Designing a Policy Provider

Drawing on our experience with identity management systems, we adopt a model in which data and policies are stored with a user-trusted entity called the policy

[6]Preventing attribute inference is related to the problem of privacy-conscious information disclosure. There has been extensive work on this subject in the context of databases, culminating in the notion of differential privacy [23] for database queries—the goal of which is to prevent an adversary from inferring attributes that could not be deduced from previously available information.

provider. A policy provider acts as a proxy for values sent by a user client:

(1) It stores values and policies.

(2) It is responsible for associating the appropriate use-based privacy policy with stored attributes.

(3) It might be responsible for ensuring that attributes are distributed and used only in compliance with the associated policies (e.g., using one of the schemes described in Chapter 4).

Policy providers implement policy-based attribute release. They release attributes only to invokers authorized by the associated Avenance policies and only for the purposes and executables specified by those policies. Released values are tagged with a policies; a service provider who receives a tagged value from a policy provider may only use that attribute in compliance with the associated policy tag.

Policy providers, which store attributes for many users, also have the potential to analyze data and detect correlated attributes; they could potentially tag raw values with policies that ensure that sensitive attributes inferred from the raw values will be associated with the appropriate policies. For example, a policy provider might detect a correlation between age and graduation date that might allow service providers to use graduation date as a *proxy* for age; if a use-based privacy policy defines restrictions on how age may be used, the policy provider might associate use restrictions with graduation date.

In the context of use-based authorizations, there is no reason to trust policy providers not to collude. We therefore envision a decentralized architecture including many, independently operated policy providers (managed by different parties) from which users can choose. Each policy provider defines or imports one or more

namespaces, thus defining the set of labels that can be used to express policies stored with that policy provider. The choice of namespace therefore imposes restrictions on what policies can be expressed.

5.3 Implementing a Policy Provider for Ohmage

We implemented a policy provider as an extension of an existing mobile health platform called Ohmage [60, 67]. Ohmage is an open-source system designed to facilitate distributed data collection and analysis for health studies and applications—an ideal candidate for use-based authorizations. It has been used for dozens of real-world studies and also serves as the backend for several production applications. Ohmage is designed with a classic three-tiered architecture, comprising a back-end database, a server component implemented in the Spring Boot framework [66], and a family of front-end mobile applications. Our policy provider is implemented in 7634 lines of Java on top of the existing Ohmage server. This implementation interfaces with Tir as described in Chapter 3.2; it can also be extended to interact with the SGX-based enforcement mechanisms as described in Chapter 4.

Data Store. The backend of Ohmage is a secure, Open mHealth-compliant data store that can be accessed through an API. The data store operates on *datapoints*, each comprised of header information (id, schema, time, source) and a `json`-encoded body; datapoints are classified by *schema*. The API allows operations for storing and retrieving datapoints: `GET datapoints/{id}`, `GET datapoints`, `POST datapoints`, and `GET datapoints/scope`. We extend the Ohmage data store to store tagged values and enforce policy tags by storing values as datapoints in Ohmage and storing tagged policies in a local MySQL database.

Policy Association. Our implementation supports both discretionary (data-subject defined) policies and mandatory (admin defined) policies through a new `POST policy` API call, which allows data subjects to modify the policy for their own datapoints and allows admins to modify the mandatory policies applied to all stored datapoints. The API has operations to modify the policy for a single specified datapoint or update the set of *preference rules*—policies that apply to all future incoming datapoints that match the specified schema. Requests to store datapoints can also specify an existing policy using the optional HTTP header `AvPolicy`.

Policy Granularity. Avenance policies could be associated with atomic values (e.g., integers) or with structured values (e.g., health records) under control of a single principal; policies could also be associated with aggregate objects containing information about many different users. Our policy provider enforces policies at the granularity of individual datapoints—in which case a request for multiple datapoints returns only the authorized datapoints—or at the granularity of datasets—in which a request for multiple datapoints is authorized only if all requested datapoints are authorized. The granularity can be configured at runtime using the command line argument `--GR.MODE`.

CHAPTER 6

RELATED WORK

Use-based Privacy. Use-based privacy was first introduced by Cate [14] as a solution to the problems presented by "notice and consent" and the underlying guidelines—the Fair Information Practice Principles [18]—that defined acceptable standards for handling sensitive data. Observing that users rarely make use of either opt-ins or opt-outs and typically don't make informed decisions about access to their data, Cate proposed a new approach. His work explored the legal and philosophical implications of use-based privacy; the feasibility of a technical regime for expressing or enforcing use-based privacy was not addressed.

Alternate Privacy Regimes. Many systems have been developed with the goal of expressing and enforcing privacy. However, none were intended to formalize, express, or enforce use-based privacy. Alternate approaches either focus exclusively on private information transmission rather than controlling usage as information flows through a networked information system (e.g., [47, 23, 53, 55]) or are unable to express the full range of reactive policies required for use-based privacy (e.g., [24, 63]).

Contextual Integrity [47] is a philosophical approach to privacy that has been formalized as a logic for reasoning about privacy [2]. Because contextual integrity defines privacy relative to socially-determined informational norms, contextual integrity can be interpreted as a special case of use-based privacy that focuses on data collection and data sharing. Transmissions are authorized when they occur in an appropriate context, as determined by social norms. The emphasis on a societal determination of acceptable or non-harmful uses (rather than informed

consent or data minimization) is closely aligned with the philosophy of use-based privacy. However, the exclusive focus on data transmission limits the applicability of existing formalizations. And although this approach supports a limited form of reactive policies based on contextual events (i.e., changes in context), it does not fulfill the full requirements for use-based privacy. In particular, this approach does not support policy synthesis for derived values, and it does not include sticky policies or obligations. It therefore cannot be used to pervasively monitor how sensitive information is used as it flows through a networked information system.

Differential privacy [23] classifies a response to a database query as a privacy violation unless the algorithm used to generate the response satisfies a specific statistical property (viz., ε-differential privacy). This definition has been formalized and implemented as an extensible platform for privacy-preserving data analysis [43]. However, differential privacy, like contextual integrity, focuses exclusively on defining authorized transmissions. Although this approach can be interpreted as a limited form of reactive policies (raw data cannot be transmitted, ε-differentially private derived values can be transmitted), it does not fulfill the full requirements for use-based privacy. In particular, this approach does not support general policy synthesis for derived values, and it does not include contextual events, sticky policies, or obligations. So like contextual integrity, it cannot be used to pervasively monitor how sensitive information is used as it flows through a networked information system.

Usage Control (UCON) [53, 54] is an extension of traditional access control models (e.g., discretionary access control, mandatory access control, role-based access control) that enables continuity of access decisions. Initial UCON systems enforced policies on a single system, but later versions introduced distributed usage

control [57, 31]. UCON can be interpreted as supporting a limited form of reactive policies—those with only contextual events—since access decisions can depend on context (e.g., time) and/or mutable attributes (e.g., number of previous access) and access control decisions are re-evaluated after the context changes. Despite the name, however, UCON does not allow access decisions to depend on the type of use. Moreover, it does not support policy synthesis for derived values. UCON, therefore, cannot effectively restrict how sensitive information is used as it flows through a system.

An alternative privacy approach was outlined by Petković et al. [55], who consider a restricted form of use-based authorization, which they call *purpose control*. Their work creates an audit log of service provider actions and then detects policy violations by checking whether the audit trail is a valid execution of the organizational process—modeled as a formula in the Calculus of Orchestration of Web Services (COWS)—for a permitted purpose. This work does not consider the invoker or the program type. Moreover, this approach does not support policy synthesis for derived values. Purpose control, therefore, cannot effectively restrict how sensitive information is used as it flows through a system.

Datta et al. [20] propose an alternative approach termed *use privacy*, which restricts the use of protected information types and their *proxies*—correlated and causally related data types. Although there is no support for reactive policies, the restrictions on proxies fulfill a similar role to history-dependent authorizations in limiting how information (rather than merely values) can be used. Their work gives an algorithm for detecting proxy use in data-driven systems (e.g., machine learning systems) and for eliminating "inappropriate" proxy uses. Although general use-based restrictions are beyond the scope of this work, their approach effectively

restricts information use by a single centralized system.

Downgrading policies [17, 40] are data use policies that specify how data should be used prior to declassification, under what conditions declassification is permitted, and how data should be treated after declassification. Conditions might include environmental events or provenance. Downgrading policies are expressive; they also offer formal guarantees that can be enforced with a language type system. However, downgrading policies lack a flexible means to express how policies might depend on sequences of events.

Thoth [24] is a kernel-level compliance layer that tracks data flow through a system and enforces declarative data use policies. The Thoth policy language specifies data use policies in terms of confidentiality, integrity, and declassification policies, each of which defines which principals are authorized and under what conditions. Although policies are designed to be expressed at a lower level than under our approach, Thoth's conditions are sufficiently flexible to capture policies that depend on who, what, or why as well as temporal, discretionary, autocratic, and jurisdictional policies. The language does not, however, support reactive policies and is therefore ill-suited to use-based privacy.

Grok [63] is a system that uses compile-time information flow analysis to enforce privacy compliance in the Bing search engine. Privacy policies are expressed in Legalease, a privacy policy language that implicitly supports use-based policies encoded as domain-specific attributes. For example, a legalese policy might say, "DENY DataType IPAddress, UseForPurpose Advertising EXCEPT ALLOW DataType IPAddress:Truncated"; this policy states that the full IP address may not be used for advertising. Grok automatically maps code-level schema elements to datatypes, minimizing the need for programmer annotations. However, Legalease

does not support reactive policies that depend on contextual events, including obligations. It also assumes that policies are defined by a centralized authority for enforcement in a centralized system; it does not support compound policies defined by multiple policies or policy synthesis for derived values whose inputs have different policies. Legalease is therefore insufficient to express the full range of use-based privacy policies.

Automata Policies. Avenance policies use privacy automata as policy rules to instantiate the reactive approach to use-based privacy proposed in this paper. Automata have long been used to model secure systems [28] and reference monitors [61, 27]. Under these frameworks, an automaton models or monitors the execution of a single program. The automaton can be interpreted as a policy for the monitored program; acceptance by the automaton means that the program is correct (or policy-compliant).

Pardo et al. [52] propose an automata-based approach to specifying dynamic privacy policies for online social networks. This work expresses evolving policies, in which the current privacy policy is activated or deactivated by contextual events. For example, "Co-workers cannot see my posts while I am not at work". These policies are parameterized over a static privacy policy language. The proposed approach does not admit synthesis events as state transitions and provides no means to construct policies for derived values.

Policy Enforcement with SGX SGX offers a new basis for placing trust in a monitor or other program, so it is a natural tool for enabling policy enforcement in distributed systems where service providers are operated by untrusted principals. Several previous projects have explored similar ideas.

Haven [3] uses SGX to create a shielded execution environment, allowing unmodified Windows application binaries to be hosted inside SGX enabled enclaves. Applications then interface with a library version of the Windows operating system running entirely inside the enclave, reducing the dependencies on the underlying system. Moreover, Haven implements a shielding module for interfacing with components outside of the enclave, which provides access to, among other things, an encrypted and integrity protected file system. While the design of Haven places the entire OS inside an enclave—allowing for applications to be securely monitored by existing enforcement mechanisms—our approach yields a smaller trusted computing base. Our work also supports policy enforcement in distributed systems.

VC3 [62] is a system for trustworthy data analytics in the cloud; it is a MapReduce framework that uses SGX to protect sensitive data. VC3 enforces confidentiality and integrity for code and data, and it enforces verifiability of code execution; it does not support enforcement for high-level, use-based authorization.

Ryoan [33] is a distributed sandbox for performing computations on sensitive data. Ryoan uses SGX enclaves to protect data confidentiality and integrity from malicious service providers; it does not support enforcement for high-level, use-based authorization.

CHAPTER 7

CONCLUSION

To be effective, an approach to privacy needs to be compatible with modern practices for data collection, data sharing, and data use. Use-based privacy has promise; this dissertation describes an investigation into the techincal feasibility of delivering on that promise.

We identify requirements for a successful use-based privacy regime, deduce that a reactive approach is essential to expressing use-based privacy, and we vet this requirement with a user study. We also introduce Avenance policies, a language that implements a reactive instantiation of use-based privacy. We evaluate the expressiveness of this instantiation by expressing the full set of data use policies defined in HIPAA and in Facebook's site privacy policy as Avenance policies.

Observing that adversaries are service providers who often have non-technical incentives to ensure policy compliance, we describe tools for facilitating policy compliance by separating policy from code and by enabling automated compliance checking. But since service providers might not be benign, we also investigate the feasibility of using Intel SGX as a root of trust to enforce policies in the presence of adversarial service providers. We describe trade-offs between deployability, security, and performance, and we demonstrate that compliance can be achieved with moderate performance overhead.

Finally, drawing on lessons from identity management systems, we describe a policy provider for associating policies with sensitive values.

We view our results as evidence of the feasibility of a reactive approach to use-based privacy. Widespread adoption will likely require further investigation into

the tradeoffs of various mechanisms for expressing and enforcing use-based privacy in real-world systems, but we view this work a promising step towards developing a privacy-enhancing, use-based privacy ecosystem for the modern world.

APPENDIX A

USER STUDY

What follows is the verbatim text of the user study we ran on May 29, 2018 to assess whether a reactive language would be well suited to collective user preferences.

Survey Instructions We are researchers trying to understand what people's priorities are regarding information privacy and how the private information is used. For each question, indicate your preferences.

Section 1: General Questions

1. Have you every adjusted your privacy preferences on Facebook or a similar site:

   (a) Never (b) At least once (c) Often (d) I don't have an account

2. Consider your preferences for how companies or websites should use your personal data. In general, might your privacy preferences change if:

   - The data were coarsened (for example, your location was defined as the city you are currently in instead of street address you are currently at):

     (a) Yes (b) No (c) Not sure

   - Only aggregate statistics were used (for example, the application only used the most popular locations across all its users, not your location individually or the application only used the average age of a user, not your personal age):

     (a) Yes (b) No (c) Not sure

- Only aggregate statistics were used (for example, the application only used the most popular locations across all its users, not your location individually or the application only used the average age of a user, not your personal age):
  (a) Yes (b) No (c) Not sure
- Time had passed (for example, only information that was more than 2 years old was used. (a) Yes (b) No (c) Not sure

3. Are you reading the questions and making an effort to answer them honestly?
   (a) Yes (b) No (c) Not sure
4. Current privacy controls online rely on notice and consent: Each website posts its privacy policy, and by using the site you agree to the terms in the policy. How often do you read these policies?
   (a)Never (b) Occasionally (c) I look at all of them (d) I read them carefully.
5. Some privacy experts have suggested replacing notice and consent with industry-wide requirements preventing harmful uses (as determined by user studies or privacy experts). How comfortable would you be with that alternative?
   (a) Very uncomfortable (b) Somewhat uncomfortable
   (c) Somewhat comfortable (d) Very comfortable

Section 2: Example Application—Medical Information

1. Assume that your hospital or another organization that stores and accesses your medical information is trying to decide on a new data privacy policy. How comfortable would you be with each of the following proposed policies:

(a) Very uncomfortable (b) Somewhat uncomfortable

(c) Somewhat comfortable (d) Very comfortable

- My medical information may be used in any way by anyone for any purpose.
- My medical information may be used to provide medical care to me.
- My medical information may be used for marketing or advertising.
- My medical information may be used to conduct medical research.
- My medical information may be used to conduct medical research if I am notified of the use and the purpose of the research.
- Anonymous versions of my medical information may be used in any way by anyone for any purpose.
- Anonymous versions of my medical information may be used to provide medical care to me.
- Anonymous versions of my medical information may be used for marketing or advertising.
- Anonymous versions of my medical information may be used to conduct medical research.
- Anonymous versions of my medical information may be used to conduct medical research if I am notified of the use and the purpose of the research.
- Aggregate statistics derived from my medicial information and that of others may be used in any way by anyone for any purpose.
- Aggregate statistics derived from my medicial information and that of others may be used to provide medical care.

- Aggregate statistics derived from my medicial information and that of others may be used for marketing or advertising.
- Aggregate statistics derived from my medicial information and that of others may be used to conduct medical research.
- Aggregate statistics derived from my medicial information and that of others may be used to conduct medical research if I am notified of the use and the purpose of the research.

2. Are there particular conditions or circumstances that might change your answers?
3. Are you reading the questions and answering them honestly?
4. Is there some policy other you would like the hospital to follow regarding your information?

Section 3: Example Application—Social Network Data

1. Assume that a social network you use (e.g., Facebook) is trying to decide on a new data privacy policy. How comfortable would you be with each of the following proposed policies: (a) Very uncomfortable (b) Somewhat uncomfortable (c) Somewhat comfortable (d) Very comfortable
   - My posts may be publicly shown.
   - My posts may be shared with friends.
   - My posts may be used to recommend posts and events I might like.
   - My posts may be used to recommend third-party apps (e.g., games) that I might like.

- My posts may be used to select ads I might be interested in.
- My posts may be used to conduct research.
- Anonymous versions of my posts may be publicly shown.
- Anonymous versions of my posts may be shared with friends.
- Anonymous versions of my posts may be used to recommend posts and events I might like.
- Anonymous versions of my posts may be used to recommend third-party apps (e.g., games) that I might like.
- Anonymous versions of my posts may be used to select ads I might be interested in.
- Anonymous versions of my posts may be used to conduct research.
- Aggregate statistics about my posts may be publicly shown.
- Aggregate statistics about my posts may be shared with friends.
- Aggregate statistics about my posts may be used to recommend posts and events I might like.
- Aggregate statistics about my posts may be used to recommend third-party apps (e.g., games) that I might like.
- Aggregate statistics about my posts may be used to select ads I might be interested in.
- Aggregate statistics about my posts may be used to conduct research.

2. Are there particular conditions or circumstances that might change your answers?
3. Are you reading the questions and answering them honestly?

4. Is there some other policy you would like the social network application to follow regarding your information?

Section 4: Demographic Information

1. Where are you from (nationality, current residence, or whichever region you identify with most):

   (a) United States (b) North America (non-US) (c) South/Central America (d) European Union (e) Europe (non-EU) (f) Asia (g) Africa (h) Australia/Pacific Islands

2. What is your age?

   (a) Younger than 18 (b) 18-22 (c) 23-35 (d) 36-65 (e) Older than 65

3. What is your gender?

   (a) Male (b) Female (c) Decline to state/Do not identify with a binary gender

APPENDIX B

ENCODING HIPAA AS DATA-USE TUPLES

Provision	Object	Invoker	Action	Purpose	Condition
§164.502(a)(1)(i)	protected health information	covered entity	disclose to individual		
§164.502(a)(2)(ii)	protected health information	covered entity	disclose		when required by the Secretary under subpart C of part 160 of this subchapter to investigate or determine the covered entitys compliance with this subpart
§164.502(d)(1)	protected health information	covered entity	use	to create information that is not individually identifiable health information	
	protected health information	covered entity	disclose to a business associate	to create information that is not individually identifiable health information	
§164.502(d)(2)	de-identified information				until reidentified
§164.502(e)(1)(i)	protected health information	covered entity	disclose to a business associate		if the covered entity obtains satisfactory assurance that the business associate will appropriately safeguard the information
§164.502(e)(3)(ii)(A)	record of disclosure of protected health information about an unemancipated minor	covered entity	disclose or provide access to a parent, guardian, or other person acting in loco parentis		if, and to the extent, permitted or required by an applicable provision of State or other law, including applicable case law

Provision	Object	Invoker	Action	Purpose	Condition
§164.506(c)(1)	protected health information (except psychotherapy notes)	covered entity	use or disclose	for its own treatment, payment, or health care operations.	
§164.506(c)(2)	protected health information (except psychotherapy notes)	covered entity	disclose	for treatment activities of a health care provider	
§164.506(c)(3)	protected health information (except psychotherapy notes)	covered entity	disclose to another covered entity or a health care provider	for the payment activities of the entity that receives the information	
§164.506(c)(4)(i)	protected health information (except psychotherapy notes)	covered entity that has or had a relationship with the individual who is the subject	disclose to another covered entity that has or had a relationship with the individual who is the subject	for a purpose listed in paragraph (1) or (2) of the definition of health care operations (other than marketing)	
§164.506(c)(4)(ii)	protected health information (except psychotherapy notes)	covered entity that has or had a relationship with the individual who is the subject	disclose to another covered entity that has or had a relationship with the individual who is the subject	for the purpose of health care fraud and abuse detection or compliance	
§164.506(c)(5)	protected health information (except psychotherapy notes)	covered entity that participates in an organized health care arrangement	disclose to another covered entity that participates in the organized health care arrangement	for any health care operations activities of the organized health care arrangement (other than marketing)	
§164.508(a)(1)	protected health information	covered entity	use or disclose consistent with authorization		with an authorization that is valid under this section
§164.508(a)(2)(i)(A)	psychotherapy notes	originator of notes	use	for treatment	

Provision	Object	Invoker	Action	Purpose	Condition
§164.508(a)(2)(i)(B)	psychotherapy notes	covered entity	use or disclosure	for its own training programs in which students, trainees, or practitioners in mental health learn under supervision to practice or improve their skills in group, joint, family, or individual counseling	
§164.508(a)(2)(i)(C)	psychotherapy notes	covered entity	use or disclosure	to defend itself in a legal action or other proceeding brought by the individual	
§164.508(a)(3)(i)(A)	protected health information	covered entity	use or disclosure in a face-to-face communication to the individual	for marketing	
§164.508(a)(3)(i)(B)	protected health information	covered entity	use or disclose in a communication constituting a promotional gift of nominal value		
§164.510(a)(1)(i)	the individual's name, the individual's location in the covered health care provider's facility, the individual's condition described in general terms that does not communicate specific medical information about the individual, and the individual's religious affiliation	covered entity	use	to maintain a directory of individuals in its facility	provided that the individual is informed in advance of the use or disclosure and has the opportunity to agree to or prohibit or restrict the use or disclosure, except when an objection is expressed in accordance with paragraphs (a)(2) or (3) of this section

Provision	Object	Invoker	Action	Purpose	Condition
§164.510(a)(1)(ii)	the object from §164.510(a)(1)(i)	covered entity	disclose to persons who ask for the individual by name		provided that the same conditions as in §164.510(a)(1)(i) apply
	the object from §164.510(a)(1)(i)	covered entity	disclose to members of the clergy		provided that the same conditions as in §164.510(a)(1)(i) apply
§164.510(b)(1)(i)	the protected health information directly relevant to such persons involvement with the individuals care or payment related to the individual's health care	covered entity	disclose to a family member, other relative, or a close personal friend of the individual, or any other person identified by the individual		with the individual's agreement
	the protected health information directly relevant to such persons involvement with the individual's care or payment related to the individual's health care	covered entity	disclose to a family member, other relative, or a close personal friend of the individual, or any other person identified by the individual		if the individual is unavailable or incapacitated and covered entity believes disclosure is in individual's best interest
§164.510(b)(1)(ii)	protected health information	covered entity	use or disclose	to notify, or assist in the notification of (including identifying or locating), a family member, a personal representative of the individual, or another person responsible for the care of the individual of the individuals location, general condition, or death.	with the individual's agreement

Provision	Object	Invoker	Action	Purpose	Condition
§164.510(b)(1)(ii)	protected health information	covered entity	use or disclose	to notify, or assist in the notification of (including identifying or locating), a family member, a personal representative of the individual, or another person responsible for the care of the individual of the individuals location, general condition, or death	if the individual is unable to agree because of incapacity
	protected health information	covered entity	use or disclose to a public or private entity authorized by law or by its charter to assist in disaster relief efforts	for the purpose of coordinating with such entities the uses or disclosures permitted by paragraph (b)(1)(ii) of this section, to notify, or assist in the notification of (including identifying or locating), a family member, a personal representative of the individual, or another person responsible for the care of the individual of the individuals location, general condition, or death	to the extent that the covered entity, in the exercise of professional judgment, determines that the requirements do not interfere with the ability to respond to the emergency circumstances
	protected health information	covered entity	use or disclose to a public or private entity authorized by law or by its charter to assist in disaster relief efforts	coordinating with such entities to notify, or assist in the notification of (including identifying or locating), a family member, a personal representative of the individual, or another person responsible for the care of the individual of the individual's location, general condition, or death	to the extent that the covered entity, in the exercise of professional judgment, determines that the requirements do not interfere with the ability to respond to the emergency circumstances
§164.512(a)(1)	protected health information	covered entity	use or disclose to the extent required by law		

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(b)(1)(i)	protected health information	covered entity	disclose to a public health authority that is authorized by law to collect or receive such information	for the purpose of preventing or controlling disease, injury, or disability	
	protected health information	covered entity	disclose to an official of a foreign government agency that is acting in collaboration with a public health authority		at the direction of a public health authority
§164.512(b)(1)(ii)	reports of child abuse or neglect	covered entity	disclose to a public health authority or other appropriate government authority authorized by law to receive reports of child abuse or neglect		
§164.512(b)(1)(iii)	protected health information	covered entity	disclose to a person subject to the jurisdiction of the Food and Drug Administration (FDA) with respect to an FDA-regulated product or activity for which that person has responsibility	for the purpose of activities related to the quality, safety or effectiveness of such FDA-regulated product or activity	
§164.512(a)(1)	protected health information	covered entity	use or disclose to the extent required by law		
§164.512(a)(1)(iv)	protected health information	covered entity	disclose to person who may have been exposed to a communicable disease or may otherwise be at risk of contracting or spreading a disease or condition		if the covered entity or public health authority is authorized by law to notify such person as necessary in the conduct of a public health intervention or investigation

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(a)(1)(v)(A)	protected health information about an individual who is a member of the workforce of the employer	a covered health care provider who is a member of the workforce of such employer	disclose to an employer		
	protected health information about an individual who is a member of the workforce of the employer	covered entity who provides health care to the individual at the request of the employer	disclose to an employer		
§164.512(a)(1)(v)(B)	protected health information that consists of findings concerning a work-related illness or injury or a workplace-related medical surveillance	covered entity	disclose to an employer		
§164.512(a)(1)(v)(C)	protected health information	covered entity	disclose to an employer		the employer needs such findings in order to comply with its obligations, under 29 CFR parts 1904 through 1928, 30 CFR parts 50 through 90, or under state law having a similar purpose, to record such illness or injury or to carry out responsibilities for workplace medical surveillance
§164.512(a)(1)(v)(D)	protected health information	covered health care provider	disclose to an employer		the covered health care provider provides written notice to the individual that protected health information relating to the medical surveillance of the workplace and work-related illnesses and injuries is disclosed to the employer

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(a)(2)	protected health information	public health authority	use	for the purpose of preventing or controlling disease, injury, or disability	
§164.512(c)(1)(i)	protected health information about an individual whom the covered entity reasonably believes to be a victim of abuse, neglect, or domestic violence (except reports of child abuse or neglect)	covered entity	disclose to a government authority, including a social service or protective services agency, authorized by law to receive reports of such abuse, neglect, or domestic violence		if required by law
§164.512(c)(1)(ii)	protected health information about an individual whom the covered entity reasonably believes to be a victim of abuse, neglect, or domestic violence (except reports of child abuse or neglect)	covered entity	disclose to a government authority, including a social service or protective services agency, authorized by law to receive reports of such abuse, neglect, or domestic violence		if the individual agrees to the disclosure
§164.512(c)(1)(iii)(B)	protected health information about an individual whom the covered entity reasonably believes to be a victim of abuse, neglect, or domestic violence (except reports of child abuse or neglect)	covered entity	disclose to a government authority, including a social service or protective services agency, authorized by law to receive reports of such abuse, neglect, or domestic violence		the covered entity, in the exercise of professional judgment, believes the disclosure is necessary to prevent serious harm to the individual or other potential victims

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(c)(1)(iii)(B)	protected health information about an individual whom the covered entity reasonably believes to be a victim of abuse, neglect, or domestic violence (except reports of child abuse or neglect)	covered entity	disclose to a government authority, including a social service or protective services agency, authorized by law to receive reports of such abuse, neglect, or domestic violence		if the individual is unable to agree because of incapacity, a law enforcement or other public official authorized to receive the report represents that the protected health information for which disclosure is sought is not intended to be used against the individual and that an immediate enforcement activity that depends upon the disclosure would be materially and adversely affected by waiting until the individual is able to agree to the disclosure
§164.512(d)	protected health information	covered entity	disclose to a health oversight agency	for oversight activities authorized by law	
§164.512(e)(1)(i)	protected health information	covered entity	disclose		in response to an order of a court or administrative tribunal
§164.512(e)(1)(ii)	protected health information	covered entity	disclose		In response to a subpoena, discovery request, or other lawful process, if the covered entity receives satisfactory assurance, as described in paragraph (e)(1)(iii) of this section, from the party seeking the information that reasonable efforts have been made by such party to ensure that the individual who is the subject of the protected health information that has been requested has been given notice of the request
	protected health information	covered entity	disclose	in response to a subpoena, discovery request, or other lawful process	if the covered entity receives satisfactory assurance, as described in paragraph (e)(1)(iv) of this section, from the party seeking the information that reasonable efforts have been made by such party to secure a qualified protective order that meets the requirements of paragraph (e)(1)(v) of this section

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(f)(1)(i)	protected health information	covered entity	disclose to a law enforcement official	for law enforcement	as required by law including laws that require the reporting of certain types of wounds or other physical injuries, except for laws subject to paragraph (b)(1)(ii) or (c)(1)(i) of this section
§164.512(f)(1)(ii)(A)	protected health information	covered entity	disclose to a law enforcement official	for a law enforcement, in compliance with and as limited by the relevant requirements of: (A) a court order or court-ordered warrant, or a subpoena or summons issued by a judicial officer, (B) a grand jury subpoena, or (C) an administrative request, including an administrative subpoena or summons, a civil or an authorized investigative demand, or similar process authorized under law	provided that: (1) the information sought is relevant and material to a legitimate law enforcement inquiry, (2) the request is specific and limited in scope to the extent reasonably practicable in light of the purpose for which the information is sought, and (3) deidentified information could not reasonably be used
§164.512(f)(1)(ii)(B)	protected health information	covered entity	disclose to a law enforcement official	for law enforcement	in compliance with and as limited by the relevant requirements of a grand jury subpoena
§164.512(f)(1)(ii)(C)	protected health information	covered entity	disclose to a law enforcement official	for law enforcement, in compliance with and as limited by the relevant requirements of an administrative request, including an administrative subpoena or summons, a civil or an authorized investigative demand, or similar process authorized under law	provided that: (1) the information sought is relevant and material to a legitimate law enforcement inquiry, (2) the request is specific and limited in scope to the extent reasonably practicable in light of the purpose for which the information is sought, and (3) deidentified information could not reasonably be used

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(f)(2)	name and address	covered entity	disclose to a law enforcement official	for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person	in response to a law enforcement official's request
	date and place of birth	covered entity	disclose to a law enforcement official	for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person	in response to a law enforcement official's request
	social security number	covered entity	disclose to a law enforcement official	for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person	in response to a law enforcement official's request
	ABO blood type and rh factor	covered entity	disclose to a law enforcement official	for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person	in response to a law enforcement official's request
	type of injury	covered entity	disclose to a law enforcement official	for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person	in response to a law enforcement official's request
	date and time of treatment	covered entity	disclose to a law enforcement official	for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person	in response to a law enforcement official's request
	date and time of death, if applicable	covered entity	disclose to a law enforcement official	for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person	in response to a law enforcement official's request
	description of distinguishing physical characteristics, including height, weight, gender, race, hair and eye color, presence or absence of facial hair (beard or mustache), scars, and tattoos	covered entity	disclose to a law enforcement official	for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person	in response to a law enforcement official's request
§164.512(f)(3)(i)	protected health information about an an individual who is or is suspected to be a victim of a crime	covered entity	disclose to a law enforcement official	for law enforcement	in response to a law enforcement official's request, if the indvidual agrees

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(f)(3)(ii)	protected health information about an individual who is or is suspected to be a victim of a crime	covered entity	disclose to a law enforcement official	for law enforcement	in response to a law enforcement official's request and if the covered entity is unable to obtain the individual's agreement because of incapacity or other emergency circumstance, provided that: (A) the law enforcement official represents that such information is needed to determine whether a violation of law by a person other than the victim has occurred, and such information is not intended to be used against the victim, (B) the law enforcement official represents that immediate law enforcement activity that depends upon the disclosure would be materially and adversely affected by waiting until the individual is able to agree to the disclosure, and (C) the disclosure is in the best interests of the individual as determined by the covered entity, in the exercise of professional judgment
§164.512(f)(4)	protected health information about an individual who has died	covered entity	disclose to a law enforcement official	alerting law enforcement of the death of the individual	if the covered entity has a suspicion that such death may have resulted from criminal conduct
§164.512(f)(5)	protected health information that the covered entity believes in good faith constitutes evidence of criminal conduct that occurred on the premises of the covered entity	covered entity	disclose to a law enforcement official	for law enforcement	

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(f) (6)(i)(A)	protected health information	covered entity	disclose to a law enforcement official	for a law enforcement	
§164.512(f) (6)(i)(B)	protected health information	covered entity	disclose to a law enforcement official	for a law enforcement	
§164.512(f) (6)(i)(C)	protected health information	covered entity	disclose to a law enforcement official	for a law enforcement	
§164.512(g)(1)	protected health information	covered entity	disclose to a coroner or medical examiner	for the purpose of identifying a deceased person, determining a cause of death, or other duties as authorized by law	
	protected health information	covered entity that also performs the duties of a coroner or medical examiner	use	for the purpose of identifying a deceased person, determining a cause of death, or other duties as authorized by law	
§164.512(g)(2)	protected health information	covered entity	disclose to funeral directors		consistent with applicable law, as necessary to carry out their duties with respect to the decedent
	protected health information	covered entity	disclose to funeral directors		if necessary for funeral directors to carry out their duties and in reasonable anticipation of, the individuals death
§164.512(h)	protected health information	covered entity	disclose to organ procurement organizations or other entities engaged in the procurement, banking, or transplantation of cadaveric organs, eyes, or tissue	for the purpose of facilitating organ, eye or tissue donation and transplantation	

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(i)(1)(i)	protected health information	covered entity	use or disclose	for research	provided that: the covered entity obtains documentation that an alteration to or waiver, in whole or in part, of the individual authorization required by 164.508 for use or disclosure of protected health information has been approved by either: (A) an Institutional Review Board (IRB), or (B) a privacy board
§164.512(i)(1)(ii)	protected health information	covered entity	use or disclose	for research	provided that the covered entity obtains from the researcher representations that: (A) use or disclosure is sought solely to review protected health information as necessary to prepare a research protocol or for similar purposes preparatory to research, (B) no protected health information is to be removed from the covered entity by the researcher in the course of the review, and (C) the protected health information for which use or access is sought is necessary for the research purposes
§164.512(i)(1)(iii)	protected health information about a decedent	covered entity	use or disclose	for research	provided that the covered entity obtains from the researcher: (A) representation that the use or disclosure sought is solely for research on the protected health information of decedents, (B) documentation, at the request of the covered entity, of the death of such individuals, and (C) representation that the protected health information for which use or disclosure is sought is necessary for the research purposes

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(j) (1)(i)	protected health information	covered entity	use or disclose to a person or persons the covered entity believes, in good faith, to be reasonably able to prevent or lessen the threat, including the target of the threat		if the covered entity, in good faith, believes the use or disclosure is necessary to prevent or lessen a serious and imminent threat to the health or safety of a person or the public
§164.512(j) (1)(ii)	protected health information (only statement and (f)(2)(i) information) not learned in course of treatment to affect propensity to commit the criminal conduct or counseling or therapy or through a requet by the individual to initiate to be referred for treatment, counseling,or therapy	covered entity	use or disclose		if the covered entity, in good faith, believes the use or disclosure is necessary for law enforcement authorities to identify or apprehend an individual: (A) because of a statement by an individual admitting participation in a violent crime that the covered entity reasonably believes may have caused serious physical harm to the victim, or (B) where it appears from all the circumstances that the individual has escaped from a correctional institution or from lawful custody
§164.512(k)(1)(i)	protected health information of individuals who are armed forces personnel	covered entity	use or disclose	for activities deemed necessary by appropriate military command authorities to assure the proper execution of the military mission	if the appropriate military authority has published by notice in the federal register the following information: (A) appropriate military command authorities, and (B) the purposes for which the protected health information may be used or disclosed

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(k)(1)(ii)	protected health information of a member of the armed forces	covered entity that is a component of the Departments of Defense or Transportation	disclose to the Department of Veterans Affairs (DVA)	for the purpose of a determination by DVA of the individual's eligibility for or entitlement to benefits under laws administered by the Secretary of Veterans Affairs	upon the separation or discharge of the individual from military service
§164.512(k)(1)(iii)	protected health information	covered entity that is a component of the Department of Veterans Affairs	use or disclose to components of the Department that determine eligibility for or entitlement to, or that provide, benefits under the laws administered by the Secretary of Veterans Affairs		
§164.512(k)(1)(iv)	protected health information of foreign military personnel	covered entity	use or disclose to their appropriate foreign military authority	for activities deemed necessary by appropriate military command authorities to assure the proper execution of the military mission	
§164.512(k)(2)	protected health information	covered entity	disclose to authorized federal officials	for the conduct of lawful intelligence, counter-intelligence, and other national security activities authorized by the National Security Act (50 U.S.C. 401, et seq.) and implementing authority (e.g., Executive Order 12333)	
§164.512(k)(3)	protected health information	covered entity	disclose to authorized federal officials	for the provision of protective services to the President or other persons authorized by 18 U.S.C. 3056, or to foreign heads of state or other persons authorized by 22 U.S.C. 2709(a)(3), or to for the conduct of investigations authorized by 18 U.S.C. 871 and 879.	

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(k)(4)	protected health information	covered entity that is a component of the Department of State	use	to make medical suitability determinations	
§164.512(k)(4)(i)	whether the individual was determined to be medically suitable	covered entity that is a component of the Department of State	disclose to the officials in the Department of State who need access to such information for the purpose of a required security clearance conducted pursuant to Executive Orders 10450 and 12698		
§164.512(k)(4)(ii)	whether the individual was determined to be medically suitable	covered entity that is a component of the Department of State	disclose to the officials in the Department of State who need access to such information as necessary to determine worldwide availability or availability for mandatory service abroad under sections 101(a)(4) and 504 of the Foreign Service Act		
§164.512(k)(4)(iii)	whether or not the individual was determined to be medically suitable	covered entity that is a component of the Department of State	disclose to the officials in the Department of State who need access to such information for a family to accompany a Foreign Service member abroad, consistent with section 101(b)(5) and 904 of the Foreign Service Act		

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(k)(5)	protected health information about an inmate	covered entity	disclose to a correctional institution or a law enforcement official having lawful custody of an inmate or other individual	for any purpose for which such protected health information may be disclosed	if the correctional institution or such law enforcement official represents that such protected health information is necessary for: (A) the provision of health care to such individuals, (B) the health and safety of such individual or other inmates, (C) the health and safety of the officers or employees of or others at the correctional institution, (D) the health and safety of such individuals and officers or other persons responsible for the transporting of inmates or their transfer from one institution, facility, or setting to another, (E) law enforcement on the premises of the correctional institution, and (F) the administration and maintenance of the safety, security, and good order of the correctional institution
§164.512(k)(6)(i)	protected health information relating to eligibility for or enrollment in the health plan	a government program providing public benefits	disclose to another agency administering a government program providing public benefits		if the sharing of eligibility or enrollment information among such government agencies or the maintenance of such information in a single or combined data system accessible to all such government agencies is required or expressly authorized by statute or regulation
§164.512(k)(6)(ii)	protected health information relating to the program	covered entity that is a government agency administering a government program providing public benefits	disclose to another covered entity that is a government agency administering a government program providing public benefits		if the programs serve the same or similar populations and the disclosure of protected health information is necessary to coordinate the covered functions of such programs or to improve administration and management relating to the covered functions of such programs.

Provision	Object	Invoker	Action	Purpose	Condition
§164.512(l)	protected health information	covered entity	disclose		as authorized by and to the extent necessary to comply with laws relating to workers' compensation or other similar programs, established by law, that provide benefits for work-related injuries or illness without regard to fault
§164.514(e)(1)	a limited data set that meets the requirements of paragraphs (e)(2) of this section	covered entity	use or disclose	research, public health, or health care operations	if the covered entity enters into a data use agreement with the limited data set recipient, in accordance with paragraph (e)(4) of this section
§164.514(f)(1)(i)	demographic information relating to an individual	covered entity	use or disclose to a business associate or to an institutionally related foundation	raising funds for its own benefit	a statement required by §164.520(b)(1)(iii)(B) is included in the covered entity's notice, (ii) the covered entity must include in any fundraising materials it sends to an individual under this paragraph a description of how the individual may opt out of receiving any further fundraising communications, (iii) the covered entity must make reasonable efforts to ensure that individuals who decide to opt out of receiving future fundraising communications are not sent such communications
§164.514(f)(1)(ii)	dates of health care provided	covered entity	use or disclose to a business associate or to an institutionally related foundation	for the purpose of raising funds for its own benefit	(i) a statement required by §164.520(b)(1)(iii)(B) is included in the covered entitys notice, (ii) the covered entity must include in any fundraising materials it sends to an individual under this paragraph a description of how the individual may opt out of receiving any further fundraising communications, (iii) the covered entity must make reasonable efforts to ensure that individuals who decide to opt out of receiving future fundraising communications are not sent such communications

Provision	Object	Invoker	Action	Purpose	Condition
§164.514(g)	protected heath information	covered entity	use or disclose	creation, renewal, or replacement of a contract of health insurance or health benefits	only as required by law
§164.522(a)(1)(iii)	protected health information	covered entity	use or disclose		if CE agrees to restriction (a)(1)(i)
	protected health information	covered entity	use		if CE agrees to restriction (a)(1)(i), and if the individual who requested the restriction is in need of emergency treatment and the restricted protected health information is needed to provide the emergency treatment
	protected health information	covered entity	disclose to a health care provider	to provide such treatment to the individual	if CE agrees to restriction (a)(1)(i) and if the individual who requested the restriction is in need of emergency treatment and the restricted protected health information is needed to provide the emergency treatment and the covered entity must request that such health care provider not further use or disclose the information
§164.524(a)(1)	protected health information (except psych. notes and info compiled in for use in a civil, criminal, or administrative action)	covered entity	disclose to the individual		no later than 30 days after receipt of the request
§164.526(a)(1)	protected health information	covered entity	amend		on request by the individual, no later than 60 days after receipt of such a request
§164.528(a)(1)	accounting of disclosures	covered entity	disclose to the individual		no later than 60 days after receipt of such a request

APPENDIX C

ENCODING FACEBOOK'S PRIVACY POLICY AS DATA-USE TUPLES

Provision	Object	Invoker	Action	Purpose	Condition
We are able to deliver our Services, personalize content, and make suggestions for you by using this information to understand how you use and interact with our Services and the people or things youre connected to and interested in on and off our Services.	things you do and information you provide, things others do and information they provide, your networks and connections, information about payments, device information, information from websites and apps that use our Services, information from third-party partners, and information from Facebook companies	Facebook	create model of how you use and interact with our Services and the people or things youre connected to and interested in on and off our Services		
	model of how you use and interact with our Services and the people or things youre connected to and interested in on and off our Services	Facebook		deliver our Services	
	model of how you use and interact with our Services and the people or things youre connected to and interested in on and off our Services	Facebook		personalize content	
	model of how you use and interact with our Services and the people or things youre connected to and interested in on and off our Services	Facebook		make suggestions for you	
We also use information we have to provide shortcuts and suggestions to you. For example, we are able to suggest that your friend tag you in a picture by comparing your friend's pictures to information we've put together from your profile pictures and the other photos in which you've been tagged.	information we have	Facebook		provide shortcuts and suggestions to you	

Provision	Object	Invoker	Action	Purpose	Condition
When we have location information, we use it to tailor our Services for you and others, like helping you to check-in and find local events or offers in your area or tell your friends that you are nearby.	location information	Facebook		tailor our Services for you and others	
We conduct surveys and research, test features in development, and analyze the information we have to evaluate and improve products and services, develop new products or features, and conduct audits and troubleshooting activities.	the information we have	Facebook	analyze	to evaluate and improve products and services, develop new products or features, and conduct audits and troubleshooting activities.	
We use your information to send you marketing communications, communicate with you about our Services and let you know about our policies and terms.	your information	Facebook	communicate with you	marketing	
	your information	Facebook	communicate with you	to communicate about our Services	
	your information	Facebook	communicate with you	to let you know about our policies and terms	
We also use your information to respond to you when you contact us.	your information	Facebook		to respond to you	when you contact us
We use the information we have to improve our advertising and measurement systems so we can show you relevant ads on and off our Services and measure the effectiveness and reach of ads and services.	the information we have	Facebook	use to improve our advertising and measurement systems	so we can show you relevant ads on and off our Services	
	the information we have	Facebook	use to measure the effectiveness and reach of ads and services		

Provision	Object	Invoker	Action	Purpose	Condition
We use the information we have to help verify accounts and activity, and to promote safety and security on and off of our Services, such as by investigating suspicious activity or violations of our terms or policies.	the information we have	Facebook		to help verify accounts and activity	
	the information we have	Facebook		to promote safety and security on and off of our Services	
We use cookies and similar technologies to provide and support our Services and each of the uses outlined and described in this section of our policy.	cookies and similar technologies	Facebook		to provide and support our Services and each of the uses outlined and described in this section of our policy	
When you share and communicate using our Services, you choose the audience who can see what you share. For example, when you post on Facebook, you select the audience for the post, such as a customized group of individuals, all of your Friends, or members of a Group.	information you share	Facebook	share with people permitted by preferences		until preference settings change
When you use third-party apps, websites or other services that use, or are integrated with, our Services, they may receive information about what you post or share.	what you post or share	Facebook	share with those third-party apps, websites or other services that use, or are integrated with, our Services		when you use third-party apps, websites or other services that use, or are integrated with, our Services
In addition, when you download or use such third-party services, they can access your public profile, which includes your username or user ID, your age range and country/language, your list of friends.	public profile	Facebook	share with third-party services		after you download or use such third-party services
	any information that you share with them	Facebook	share with third-party services		after you download or use such third-party services

Provision	Object	Invoker	Action	Purpose	Condition
Information collected by these apps, websites or integrated services is subject to their own terms and policies.	information collected by third-party apps	third-party services	uses permitted by their terms and policies		
We share information we have about you within the family of companies that are part of Facebook.	information we have about you	Facebook	share within the family of companies that are part of Facebook		
If the ownership or control of all or part of our Services or their assets changes, we may transfer your information to the new owner.	your information	Facebook	transfer to the new owner		if the ownership or control of all or part of our Services or their assets changes
We use all of the information we have about you to show you relevant ads.	all the information we have about you	Facebook		to show you relevant ads	
We do not share information that personally identifies you with advertising, measurement or analytics partners unless you give us permission.	information that personally identifies you	Facebook	share with advertising, measurement or analytics partners		if you give permission
We may provide these partners with information about the reach and effectiveness of their advertising without providing information that personally identifies you, or if we have aggregated the information so that it does not personally identify you.	non-personally identifying information about reach and effectiveness of their advertising	Facebook	share with advertising, measurement or analytics partners		
	aggregate information	Facebook	share with advertising, measurement or analytics partners		
We transfer information to vendors, service providers, and other partners who globally support our business. These partners must adhere to strict confidentiality obligations in a way that is consistent with this Data Policy and the agreements we enter into with them.	information	Facebook	transfer to vendors, service providers, and other partners who globally support our business		these partners must adhere to strict confidentiality obligations in a way that is consistent with this Data Policy and the agreements we enter into with them

Provision	Object	Invoker	Action	Purpose	Condition
Information associated with your account will be kept until your account is deleted, unless we no longer need the data to provide products and services.	information associated with your account	Facebook	delete		after you delete your account
	information associated with your account	Facebook	delete		when no longer needed to provide products and services
When you delete your account, we delete things you have posted, such as your photos and status updates.	things you have posted	Facebook	delete		after you delete your account
We may access, preserve and share your information in response to a legal request (like a search warrant, court order or subpoena) if we have a good faith belief that the law requires us to do so.	your information	Facebook	access, preserve and share	to response to a legal request	if we have a good faith belief that the law requires us to do so
We may also access, preserve and share information when we have a good faith belief it is necessary to: detect, prevent and address fraud and other illegal activity; to protect ourselves, you and others, including as part of investigations; or to prevent death or imminent bodily harm.	your information	Facebook	access, preserve and share		when we have a good faith belief it is necessary to detect, prevent and address fraud and other illegal activity
	your information	Facebook	access, preserve and share		when we have a good faith belief it is necessary to protect ourselves, you and others, including as part of investigations
	your information	Facebook	access, preserve and share		when we have a good faith belief it is necessary to prevent death or imminent bodily harm

APPENDIX D

DETAILS OF EXISTING IDENTITY MANAGEMENT SYSTEMS

This appendix discusses historical and current identity management solutions in chronological order, and it sketches design choices made by each. We discuss Passport, Project Liberty, Idemix, Shibboleth, Higgins, PRIME, OpenID, CardSpace, U-Prove, P-IMS, and Facebook Single Sign-On. The primary focus is implementations rather than specifications, but we do discuss identity management specifications for which there are no representative implementations (e.g., Project Liberty, OpenID). Specifications for which a representative implementation exists (e.g., SAML, OASIS-IMS, WS-InfoCard) are covered by the systems we discuss that implement those specifications.

D.1 Passport (1999)

Passport is an identity management system introduced by Microsoft in 1999; it is no longer actively supported. Microsoft ran the sole identity provider in the system, and service providers joined the Passport system by registering with Microsoft, at which time they received a unique SiteID and a DES encryption key. After registering, a service provider could delegate identity management and user authentication to Microsoft.

Like many early identity management systems, Passport employed interactive authentication (thereby allowing the identity provider to both detect user actions and observe the context in which those actions occur). When a user requested access to a protected resource or service, the service provider redirected the request to a Passport server (locations were periodically published) and included the SiteID

and a return URL as parameters. The Passport server verified the SiteID and returned URL against the list of registered service providers; if they matched, then Passport authenticated the user (either by acquiring a valid email-password combination and setting a DES-encrypted cookie or by reading a previously-defined cookie). Passport then encrypted the user's Passport Unique Identifier (PUID), timestamp, and public profile information under the service provider's DES key and redirected the user's browser to the return URL, with the encrypted information included as a parameter. The service provider decrypted the information and used it to make an authorization decision.

The centralized Passport identity provider stored a database of user identities. Each identity consisted of a unique identifier (PUID), authentication credentials (email, password, and an optional four-digit security key), and attributes. Nothing prevented Microsoft from linking different identities on the basis of source, usage, or attribute. Authentication assertions issued by Passport servers contained the unique PUID (so the system is pseudonymous) and, therefore, service providers could link an authorization request to the identity issuing that request, and could link two requests made with the same identity.

Passport provided minimal support for confidentiality. Since Microsoft ran the only identity provider, all attributes were stored on Microsoft servers; attributes were sent to a service provider on the basis of tags that labeled the attribute as either `public` or `private`. Attributes could be collected by Passport during registration (e.g., email address) or could be contributed by a service provider. All attributes (except PUID) were `private` by default, but a user could voluntarily set attribute tags to `public`.

Initially, the only attribute was an email address, but users and service providers

could add other attributes. Certain types of information—e.g., financial information—were tagged as *wallet attributes* by Passport. Wallet attributes were disclosed only after receiving explicit, interactive, permission from the user. Attributes were encrypted using a private, shared key, so they were deniable. However, there was no support for obligations, and there was no protection against attackers inferring attributes from user actions or from other attributes.

Passport was never widely adopted. Several theories attempt to explain why. The most prevalent concerns the limited support Passport offered for user privacy—in particular, user actions were detectable, observable, and linkable. Moreover, Microsoft, as the exclusive identity provider in the system, was in a position to take advantage of these privacy vulnerabilities. The discovery of several security vulnerabilities in the Passport implementation [36] also undermined confidence in Passport.

D.2 Project Liberty (2001)

The Liberty Alliance was a group of companies formed in 2001 to define standards that would enable individuals and businesses to engage in secure, private transactions by leveraging federated identity management systems. The standards were collectively referred to as Project Liberty, and involved three phases, released incrementally:

- **Phase 1:** Identity federation and single sign-on,
- **Phase 2:** A framework for building identity services (including attribute sharing),

- **Phase 3:** Interoperable identity services (including services to manage identities and profiles).

In 2005, the Project Liberty specifications were incorporated into SAML 2.0, a specification implemented by Shibboleth 2.0 and various enterprise identity solutions. The organization aspects of the Liberty Alliance were subsequently subsumed by the Kantara Initiative.

Since Project Liberty was a set of specifications rather than a specific implementation, it admitted multiple design choices. It specified three possible control flows that supported authentication assertions: Artifact-based, POST-based, and Liberty-Enabled Client-based. The first two approaches were interactive, while the third was a form of active-client authentication. All three approaches allowed identity providers to detect user actions and to observe the context in which those actions occur. The three specified control flows are summarized in Figure D.1.

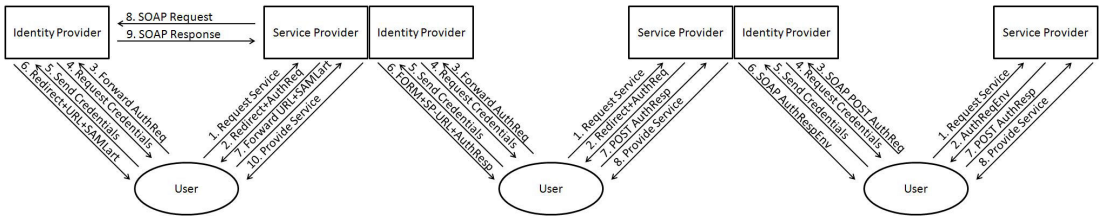


Figure D.1: Types of Delegated Authentication Specified in Project Liberty

The Liberty Alliance envisioned a world in which (1) businesses would form *circles of trust* based on contractual agreements and (2) users would have an option to link or federate local identities within an established circle of trust. If a user chose to federate an identity, then the identity provider published an *introduction* to service providers in the same circle of trust (how this was implemented was

not fixed by the Liberty Project specification, but use of a common domain was suggested). Service providers could *federate* or link a local identity with the identity stored at the identity provider using such introductions. Identities that were stored with un-federated identity providers (e.g., those in different circles of trust) could not.

By default, the authorization protocol was pseudonymous (the user is referred to by a local, shared handle), and user actions could be linked to an identity. However, the specification also included an option for the service provider to enable anonymous authorization using a temporary identifier. Whether a service provider could link two actions depended on whether the service provider opted for pseudonymous authorization; the use of an authentication credential could always be linked to its issuance.

The Project Liberty specifications allowed authentication assertions to include information about the authentication context [42] and user attributes [35]. How the system implemented user control over attribute confidentiality was beyond the scope of the Project Liberty specification, but the documentation recommended that disclosure of user attributes be governed by identity provider policies, user permissions, and interaction with the user. In the Bluewin [6] implementation of Project Liberty, users were asked to consent to attribute sharing when federating identities and when accessing the provider in question [68]. The specification did include some protection against attribute disclosure by third parties (i.e., attribute forwarding); since users were referred to by local handles, authentication responses (including any asserted attributes) were deniable. Project Liberty provided no mechanism to prevent attribute inference.

D.3 Idemix (2001)

Idemix is an anonymous credential system first proposed by researchers at IBM Zurich in 2001. Idemix is currently in a pilot phase, where it is employed in government, banking, and telecommunications; it is available as a Java library. Enhanced versions have also been used in the PRIME and Higgins projects.

All identity providers and service providers in the system generate and publish public keys that can be used for encrypting messages and verifying digital signatures. Each user U generates a secret key S_U that is used to generate pseudonyms for that user. U can establish an identity $N_{IDP}(S_U)$ that consists of attributes $\{attr_1, \ldots, attr_n\}$ with an identity provider *IDP*.

Idemix is a credential-based system, so user actions are undetectable to the identity provider. A user U obtains a credential $C(IDP, N_{IDP}, \{attr_1, \ldots, attr_n\})$ from an identity provider *IDP* by presenting authentication credentials for identity N_{IDP}. Credentials can be issued as one-time credentials, can be issued with an expiration date, or can be issued with a parameter that can be validated using a global revocation manager. To subsequently access a service provider *SP*, a user U demonstrates possession of an appropriate credential. If a local pseudonym $N_{SP}(S_U)$ has previously been established with the service provider, then U proves knowledge of a credential that incorporates the desired set of attributes and was issued to a pseudonym N' that is linked to the same secret key as the local pseudonym N_{SP}; if not, then U simply proves knowledge of a credential that incorporates the desired set of attributes and was issued to a pseudonym N' for which U knows the corresponding secret key.

Because the proof of knowledge employed during credential presentation re-

quires the presenter to know secret key S_U, the Idemix protocol can ensure that any user U' who can successfully present a credential issued to a user U can successfully use all of U's credentials. This functionality is intended to prevent users from sharing the secret key with others. The secret key S_U can also be tied to knowledge of an external secret (e.g., bank account information), providing an additional incentive for users to keep their credentials private.

Idemix is designed to minimize linking. Different identities cannot be linked, actions cannot be linked to identities (because credentials are anonymous), and it is impossible for two actions to be linked (even if a user presents the same credential multiple times and even if service providers collude with identity providers). However, in order to support accountability, a user can voluntarily allow a trusted party (the identity provider or a third-party) to link the use of a credential to a pseudonym N_{IDP} or other identifying information by encrypting it under the trusted party's secret key and providing this information (along with a proof that it was correctly generated) when a credential is presented. Since service providers make authorization decisions, a service provider can choose to authorize only those users who submit de-anonymizable credentials.

Idemix implements instance-based attribute release; a user must make individual decisions regarding which attributes are stored with each identity provider and which attributes are disclosed to which service provider. Idemix credentials do, however, support selective release of attributes. Credentials can be used to assert subsets of the attributes or to assert "coarser" versions of the attributes. For example, if the credential were issued to assert that $N.age = 25$ and $N.employer =$ *Example Co.* then it could be used assert only that $N.age \geq 18$; this is implemented by proving that a parameter in the credential is within a range rather equal

to a specific value. Although Idemix credentials are deniable (since the successful proof of an attribute or claim requires knowledge of the secret S_U), Idemix provides no mechanisms to protect the confidentiality of attributes against inference.

D.4 Shibboleth (2003, v2.0 released 2006)

Shibboleth is a standards-based, open-source software package developed by Internet2 that implements the SAML standard to provide federated single sign-on and attribute exchange; it also provides functionality to manage attribute release. The first version was released in 2003, and Shibboleth 2.0 (which implements SAML 2.0 for its core protocols) was released in 2006. It is deployed primarily in academic environments; universities function as Shibboleth identity providers, and services to which they have subscribed (e.g., JSTOR) use the Shibboleth service provider implementation to control access to protected resources.

Shibboleth implements interactive authentication. When a user attempts to access a protected resource at a service provider, the service provider first identifies identity provider(s) with which the user is affiliated. This identification is done either by interacting directly with the user (e.g., presenting a list of identity providers and asking the user to select one) or by interacting with a centralized OASIS discovery service [70]. The service provider then redirects the request to the appropriate identity provider, and the user authenticates to that identity provider; the authentication method can either be specified by the service provider or can be determined by the identity provider. After a successful authentication, the identity provider sends a signed, encrypted authentication assertion (or a reference to it) back to the service provider via the user. The service provider verifies

and unpackages the authentication assertion and uses it to make an authorization decision.

Whether Shibboleth is vulnerable to linking depends on some user settings. A user can prevent different identities from being linked (since Shibboleth is a decentralized system), or the user can choose to store distinct identities with the same identity provider. Shibboleth supports both pseudonymous and anonymous authorization (anonymous authorization can be required by any party involved in the interaction). If anonymous authorization is chosen, then actions cannot be linked to identities and authorization requests cannot be linked to each other (although the use of an authentication assertion can be linked to its issuance if the service provider and identity provider collude).

In comparison to earlier systems, Shibboleth provides sophisticated protection for attribute confidentiality. A user makes individual, itemized decisions about which attributes to store with each identity provider, and Shibboleth then supports expressive policies governing disclosure of these attributes to service providers. Which attributes are disseminated (that is, which attributes are included in an authentication assertion) is determined by an attribute filter. An attribute filter is a collection of policy rules. Each policy rule consists of a single requirement rule—which determines whether the given policy rule is applied to a particular authentication request—and zero or more attribute rules—which specify what attributes are affected by the policy rule and whether or not they will be released or whether release is contingent on explicit, interactive user consent. Policy rules can depend on requester, issuer, principal, or attribute, and each of these can depend on exact matches, matching a regular expression, format, attributes of the party in question, regular expressions for attributes of the party in question, or arbitrary

scripts. In order to minimize attribute forwarding, an identity provider may make assertions deniable by sending a reference rather than a signed assertion. However, no mechanism prevents attribute inference.

D.5 Higgins (2003, v2.0 released 2011)

Begun as an effort by Paul Trevithick to implement a dashboard for managing multiple distinct identities, Higgins developed into an open-source Information Card system which simplifies the development and implementation of identity solutions. Currently, the primary goal of Higgins is to encourage deployment of identity management systems (and the Higgins Information Card functionality, in particular) by facilitating installation and by supporting multiple platforms (including mobile systems). Higgins 1.0 implements the OASIS Identity Management interoperability protocol and performs the functionality of an identity card selector; it was released by the Eclipse Foundation in 2008. Higgins 2.0 adds support for a personal data store.

Higgins is an active-client system in which the user decides what messages are sent to which parties (currently by running a browser extension and/or native applications). After establishing an identity with an identity provider, a user receives an *information card* associated with that identity. An information card is a reference to an identity that lists the types of security tokens (e.g., x.509 certificates, Idemix credentials, etc.) and the types of attributes that can be obtained from the identity provider. Depending on the Higgins version, information cards can be stored locally or on a dedicated Higgins server. A user can always become an identity provider by creating personal or self-issued information cards for

attributes being claimed (e.g., username, address). When a user employs a particular information card, an authentication assertion corresponding to that identity is requested from the appropriate identity provider. Identity providers, therefore, detect user actions, but they cannot necessarily observe the context in which those actions occur.

Higgins is a decentralized system, so distinct identities established with distinct identity providers cannot be linked by any single party in the system (other than the user who established those identities). Whether user actions can be linked to identities or to other actions depends on the type of authentication assertion that is employed, something not specified by Higgins.

Confidentiality of attributes is controlled by an instance-based attribute release mechanism. A user explicitly decides which attributes to share with each identity provider when establishing a digital identity. Subsequently, during authorization, the service provider releases a list of requirements (specified either using WS-SecurityPolicy or HTML) and the user must decide whether to release the requested attributes and, if so, which identity to use. Once the user selects an identity (or information card), a request is sent to the corresponding identity provider, and it issues an authentication assertion for the requested attributes (in the required format, if specified). Whether and how the release of attributes across invisible channels is controlled depends on the format of the authentication assertion, which is not specified by Higgins.

D.6 PRIME (2004, v3.0 released 2008)

PRIME is the result of the four-year Privacy and Identity Management for Europe project that ran from 2004 to 2008; the system is a research prototype. The primary focus of PRIME is enabling users to effectively protect and control personal information.

PRIME assumes that users will serve as identity providers for self-asserted attributes; the system includes middleware that users can download and run for this purpose. Distinct, third-party identity providers—that independently validate attributes—are intended to issue high-assurance authentication assertions for use with service providers that do not trust self-asserted attributes.

PRIME is a credential-based identity management system. Prior to issuing an authorization request, a user interacts with zero or more identity providers to obtain authentication assertions for various attributes. The details concerning how a third-party identity provider validates attributes and authenticates users are not specified. Authentication assertions are implemented as digitally signed statements or Idemix credentials.

After receiving an authorization request, a service provider responds with a *Data Handling Policy*, which describes types of authentication assertions required for an affirmative authorization decision. The user then describes conditions under which the user will release various attributes, and PRIME automatically determines whether there exist attributes, formats, and conditions under which both parties would satisfied. The conditions could require that released attributes be tagged with obligations specifying notification requirements, deletion requirements, and other limitations on attribute use. Any obligations are automatically enforced

by PRIME.

If an agreement is reached, then the user presents an authentication assertion containing appropriate (tagged) attributes. That assertion also contains a pseudonym, which can be relationship-based, role-based, role-relationship-based, or session-based, depending on preferences of the user and the service provider. Pseudonyms can be arbitrarily assigned strings or can be Idemix pseudonyms. The degree to which linking is possible depends on the nature of the pseudonym.

D.7 OpenID (2005, v2.0 ratified 2007)

OpenID is an open standard for decentralized identity management that was developed in 2005; the specification for OpenID 2.0 adds support for attribute management and was ratified in 2007. OpenID is now provided and accepted by many websites, including Google, IBM, Yahoo!, AOL, PayPal, VeriSign, MySpace, LiveJournal, and Steam.

OpenID specifies an interactive authentication protocol. The protocol allows identity providers to detect (and observe the context of) user actions. Authentication is initiated when a user requests a protected service from a service provider and provides an OpenID identifier. Such an identifier is a URL or XRI (e.g., `username.identityprovider.com`). The endpoint for the identity provider associated with the given identifier is determined by performing XRI resolution, employing the Yadis protocol, or by retrieving a document from the specified URL. The service provider and identity provider then optionally establish a *relation* (if there is none) by exchanging secret keys. Next, the service provider redirects the request to the identity provider with an OpenID authentication request. The

identity provider authenticates the user and then redirects the request back to the service provider with a (signed) OpenID assertion. The service provider verifies the assertion and signature (using the shared secret key or via direct communication with the identity provider) and then makes an authorization decision.

The OpenID specification describes a decentralized system in which different identities associated with the same user cannot be linked. It supports both pseudonymous and anonymous modes of authorization (which method is employed is specified by the service provider). When anonymous authorization is employed, user authorization requests can be linked to a corresponding authentication assertion, but two distinct authorization requests made by the same user cannot be linked to each other.

OpenID 2.0 [9] introduces support for attributes [29], but protocols to enforce confidentiality are not specified. Attribute types are given as URIs, and attributes are arbitrary data encoded as UTF-8 strings. How attributes are created and which identity providers are given access to those attributes is not described by the specification, but it is suggested that all parties in the system (including identity providers and service provides) be allowed to create attributes associated with a user and to make itemized decisions regarding which identity providers store those attributes. A service provider can specify requested attribute types (designated either *required* or *if_available*); how an identity provider determines which attributes to release is not covered in the specification, but the Google OpenID identity provider gives the user the (interactive) option to permit or deny the release of each attribute on the required list (with an option to remember the selected permissions for future interactions with that service provider). OpenID attributes are deniable, but no other mechanisms for preventing or minimizing the release of

attributes across invisible channels are specified.

D.8 CardSpace (2006)

CardSpace is a proprietary identity management metasystem released by Microsoft in 2006 as part of the .NET Framework 3.0. It was designed to facilitate management of multiple digital identities or "identity cards" and is consistent with the OASIS IMI standard. CardSpace was designed to work with arbitrary token formats (e.g., OpenID, SAML, U-Prove), but it was compatible only with Internet Explorer run on Windows operating systems. In 2011, Microsoft announced that it would not be releasing CardSpace 2.0 and would instead focus its attention on its new U-Prove system (described in Section D.9).

CardSpace is an active-client system. A user (or a browser acting on behalf of the user) decides what messages are sent to which parties in the system. Given the limitations of current web browsers, a user must install the CardSpace client locally in order to use the identity management system.

Upon establishing an identity with an identity provider, the user is given an information card. An information card is a locally-stored reference to an identity that lists the available types of credential, the list of attributes that can be obtained with that identity, a URL for requesting credentials, a time-stamp, and a globally unique identifier; information cards are signed by the issuing identity provider. CardSpace users can also serve as identity providers for self-asserted attributes (e.g., username).

Since CardSpace is an active-client system, identity providers can detect user

actions, but they cannot necessarily observe the context in which those actions occur. The default implementation allows identity providers to observe which attributes are released at what points in time. However, the identity provider cannot observe which service provider receives those attributes. CardSpace identity providers may optionally require a service provider's identity (information which is necessary for certain types of credential).

Since CardSpace is an active-client system in which users store information locally, it must implement means to support users who use multiple devices. To address this, information cards can be copied onto a portable storage medium. However, in released versions of CardSpace, the user must download the information cards onto the local device prior to use.

CardSpace is a decentralized system. So, distinct identities established with distinct identity providers cannot necessarily be linked by any single party (other than the user who established those identities). Whether user actions can be linked to identities or to other actions depends on the type of credential that is employed and is not specified by CardSpace.

Confidentiality of attributes is controlled by an instance-based access control mechanism. A user explicitly decides which attributes to share with each identity provider when establishing a digital identity. In response to an authorization request, a service provider releases a list of requirements (specified either using WS-SecurityPolicy or HTML) and the user must decide whether to release the requested attributes and, if so, which identity to use (the CardSpace system, running locally on the user's machine, determines which identities can fulfill the specified requirements). Once a user selects an identity (or information card), a request is sent to the corresponding identity provider, and the identity provider issues a cre-

dential for the requested attributes (in the required format, if specified). Whether and how the release of attributes across invisible channels is controlled depends on the type of credential that is used and, therefore, is not specified by CardSpace.

D.9 U-Prove (2007)

U-Prove is an identity management solution based on cryptographic protocols developed by Stefan Brands [7]. First released by Credentica in 2007, U-Prove was subsequently acquired and re-branded by Microsoft. It is now available as an open-source implementation.

U-Prove is a credential-based identity management system where the credentials (called *tokens* in the U-Prove documentation) consist of digitally signed collections of attributes. More specifically, a U-Prove token consists of a unique identifier, a public key that encodes the attributes, a token information field that includes usage restrictions and expiration, the issuing identity provider's blind signature of the previous elements, and prover-asserted information (e.g., self-asserted attributes or a service provider-supplied nonce). In order to obtain a U-Prove token for a particular identity, a user authenticates with an identity provider and then participates in the interactive U-Prove issuance protocol, which returns a token and a private key (known only to the user). A U-Prove token can only be used by parties that know this private key. If token delegation and transfer are undesirable, then users can be discouraged from sharing private keys by linking the value to that of an external secret (e.g., bank account information). Since signatures and, therefore, tokens can be verified non-interactively, U-Prove is a credential-based system in which user actions are undetectable to the identity provider.

U-Prove is a decentralized identity management system. Users can, therefore, establish distinct identities with distinct identity providers, preventing those identities from being linked. U-Prove also is an anonymous authorization system, and user actions cannot be linked to the identity to which a credential was issued. Because token presentation involves sending the service provider a unique token identifier, the service provider can link different sessions in which the user was authorized using the same token. However, uses of two different tokens cannot be linked. Since U-Prove employs blind signatures to generate tokens, U-Prove tokens are untraceable (an identity provider cannot link the use of a token to the token that was issued).

U-Prove provides minimal technical support for confidentiality of user attributes. A user must make instance-based attribute release decisions to determine which attributes are disclosed to each identity provider. The mechanism used to control disclosure of attributes to service providers is not discussed in the U-Prove documentation. However, in the version integrated into CardSpace, the service provider requests a collection of attributes whenever a user wants to access a protected resource. The user responds by sending a U-Prove credential that includes the requested attributes as well as a zero-knowledge proof that the credential was generated using those attributes (and possibly other undisclosed attributes) and a private key known to the user. Observe, this allows release of subsets of the attributes in a given U-Prove credential. Attributes are deniable (since zero-knowledge proofs, by definition, cannot be reproduced), but U-Prove provides no mechanisms to prevent or control attribute inference.

D.10 P-IMS (2008)

Persona-based Identity Management (P-IMS) is an identity management system proposed by researchers at Queen's University in 2008. The primary goal of P-IMS is to find a balance between anonymity and accountability in a credential-based system. P-IMS currently exists only on paper.

P-IMS is designed to exploit the features of an identity-based signature scheme. ID-based signatures allow a party in the system to sign messages in a manner that permits other parties to verify the signature using only an identifier associated with the signer; private signing keys are issued to an identifier by a designated *key generator*. In the P-IMS system, each identity provider functions as a key generator, and the cryptographic identifiers are hashes of the (encrypted) list of attributes associated with an identity. To create an identity (which is called a *persona* in the P-IMS documentation), a user gives an identity provider a list of attributes that the user wishes to associate with that identity; the user may also optionally present credentials that attest to one or more of the claimed attributes. The identity provider validates the claimed attributes (a process that could require real-world interactions and/or verification of the presented credentials) prior to creating an identity with those attributes.

P-IMS is a credential-based authentication system, so user actions are undetectable to the identity provider. Upon receiving a request, an identity provider issues a credential for an established identity I (which is a collection of attributes along with any credentials P presented in support of those attributes) by generating the private key PK_I associated with identifier $hash(I)$ and sending $C = (I, PK_I)$ to the user. To present a credential, a user sends the signed identity (where the signature is generated using the key PK_I) or a signed list of label-encrypted attributes

(along with labels for the desired attributes) to a service provider. The latter allows the user to selectively disclose attributes from a pre-defined credential, since label-encrypted ciphertexts require both the private key and a ciphertext-specific label in order to decrypt the ciphertext. After a credential has been presented, a service provider can non-interactively verify the credential by verifying the signature using the appropriate identifier. Non-interactive verification ensures that authorization requests remain undetectable to the identity provider.

Since P-IMS is envisioned as a decentralized identity management system with many identity providers, distinct identities associated with the same user may be stored with distinct identity providers, thereby minimizing the threat of identity linking. Moreover, P-IMS is an anonymous authorization system; credentials are not labeled with a name or pseudonym, so actions cannot be inherently linked to the user behind them. However, all authorization requests that are invoked using the same credential include the same identity (or label-encrypted list of attributes). Therefore, a service provider can link two actions that were invoked using the same credential.

P-IMS provides limited support for attribute confidentiality. Attribute release to both identity provider and service provider is controlled using an instance-based approach, although label-encryption does permit a user to release subsets of the attributes asserted in a credential. No mechanisms prevent or limit communication of attributes across invisible channels: attributes are not deniable (a credential can be non-interactively verified by any party), and there is no support for obligations. There are also no mechanisms designed to prevent or minimize attribute inference.

D.11 Facebook Single Sign-On (2008, released 2010)

Facebook is commonly considered to be a service provider. However, with the introduction in 2008 of Facebook Connect—a proprietary single sign-on system—Facebook became an identity providers as well. Its current identity management system, Facebook Single Sign-On, is a component of Facebook's Open Graph platform that implements the OAuth 2.0 [49] authorization specification. By leveraging the large Facebook user base and by providing economic incentives for service providers (namely access to large quantities of user data, including the social graph), Facebook Single Sign-On has become one of the most widely-deployed identity management systems [38].

Facebook Single Sign-On implements an interactive authentication protocol specified by OAuth 2.0. When a user attempts to access a protected resource, the service provider presents a login page that includes the option to authenticate with a Facebook account. When a user selects this option, the browser opens a Facebook window that asks for authentication credentials (email address and password) and any requested permissions. Facebook also sets a session cookie at this time. By default, basic information (which includes name, picture, friends list, and any information the user has made public) is requested; applications can also request additional permissions. After a user has authenticated with Facebook and granted any requested permissions, Facebook issues an *access token*—a signed string containing agreed permissions and an expiration date—which is forwarded to the service provider. The service provider can use the access token to make requests to Facebook's APIs, and it can make authorization decisions on the basis of the received information.

Facebook Single Sign-on is a centralized identity management system: Face-

book is the sole identity provider. However, linking between multiple identities is not considered a privacy concern, since the Facebook terms of service limit a user to a single identity. Facebook Single Sign-On does not prevent linking between actions and identities or between different actions because it does not support anonymous authorization. Any service provider that receives an access token can access the user's "real-world" name.

Facebook offers little support for confidentiality of user attributes; if a user wishes to interact with a service provider using Facebook Single Sign-on then that user must grant the service provider all requested permissions. After permissions are granted, the service provider can use the resulting access token to make arbitrary calls to the Facebook APIs up until the token expires, and the resulting information can be used for any purpose.

BIBLIOGRAPHY

[1] Sruthi Bandhakavi, Charles C. Zhang, and Marianne Winslett. Super-sticky and declassifiable release policies for flexible information dissemination control. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, pages 51–58, 2006.

[2] Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *IEEE Symposium on Security and Privacy*, pages 184–198, 2006.

[3] Andrew Baumann, Marcus Peinado, and Galen Hunt. Shielding applications from an untrusted cloud with haven. In *Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation*, pages 267–283. USENIX Association, 2014.

[4] Eleanor Birrell. Avenance middleware. `https://bitbucket.org/cornell-ebirrell/av-middleware`, 2018.

[5] Eleanor Birrell and Fred B. Schneider. A reactive approach to use-based privacy. Technical Report 54843, Cornell University, Computing and Information Science, November 2017.

[6] Bluewin. `http://www.bluewin.ch`.

[7] Stefan Brands. *Rethinking public key infrastructures and digital certificates—building in privacy*. PhD thesis, Eindhoven University of Technology, September 1999.

[8] Travis D. Breaux, Matthew W. Vail, and Annie I. Anton. Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In *Requirements Engineering, 14th IEEE International Conference*, pages 49–58. IEEE, 2006.

[9] Johnny Bufu, Josh Hoyt, and David Recordon. OpenID authentication 2.0. `http://openid.net/specs/openid-authentication-2_0.html`, December 2007.

[10] Laurent Bussard, Gregory Neven, and F.-S. Preiss. Downstream usage control. In *IEEE Internation Symposium on Policies for Distributed Systems and Networks (POLICY)*, pages 22–29, 2010.

[11] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology EUROCRYPT 2001*, pages 93–118, 2001.

[12] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 21–30, 2002.

[13] Sébastien Canard, Eric Malville, and Jacques Traoré. Identity federation and privacy: One step beyond. In *Workshop on Digital Identity Management*, 2008.

[14] Fred Cate. Principles for protecting privacy. *Cato Journal*, 22:33–57, 2002.

[15] Fred Cate, Peter Cullen, and Viktor Mayer-Schönberger. Data protection principles for the 21st century. Oxford Internet Institute, 2013.

[16] David Chappell. Introducing windows cardspace. `http://msdn.microsoft.com/en-us/library/aa480189.aspx`, April 2006.

[17] Stephen Chong and Andrew C Myers. Security policies for downgrading. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 198–209, 2004.

[18] Federal Trade Commission et al. Fair information practice principles. *last modified June*, 25, 2007.

[19] Intel Corp. Intel software guard extensions (intel sgx). `https://software.intel.com/sites/default/files/332680-002.pdf`, June 2015.

[20] Anupam Datta, Matthew Fredrikson, Gihyuk Ko, Piotr Mardziel, and Shayak Sen. Use privacy in data-driven systems: Theory and experiments with machine learnt programs. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1193–1210. ACM, 2017.

[21] Docker. `http://www.docker.com/`.

[22] Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Obligations and their interaction with programs. In *European Symposium on Research in Computer Security*, pages 375–389. Springer, 2007.

[23] Cynthia Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP)*, volume 4052, pages 1–12, Venice, Italy, July 2006. Springer Verlag.

[24] Eslam Elnikety, Aastha Mehta, Anjo Vahldiek-Oberwagner, Deepak Garg, and Peter Druschel. Thoth: Comprehensive policy compliance in data retrieval systems. In *USENIX Security Symposium*, pages 637–654, 2016.

[25] Facebook. Data policy. `https://www.facebook.com/full_data_use_policy`, September 2016.

[26] Deepak Garg, Limin Jia, and Anupam Datta. Policy auditing over incomplete logs: Theory, implementation and applications. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 151–162. ACM, 2011.

[27] Richard Gay, Heiko Mantel, and Barbara Sprick. Service automata. In *Formal Aspects in Security and Trust*, volume 7140 of *Lecture Notes in Computer Science*, pages 148–163. Springer, 2011.

[28] Joseph Goguen and José Meseguer. Security policies and security models. In *Security and Privacy, 1982 IEEE Symposium on*, pages 11–11. IEEE, 1982.

[29] Dick Hardt, Johnny Bufu, and Josh Hoyt. OpenID attribute exchange 1.0. `http://openid.net/specs/openid-attribute-exchange-1_0.html`, December 2007.

[30] Higgins. `http://www.eclipse.org/higgins/`.

[31] M. Hilty, A. Pretschner, D. Basin, C. Schaefer, and T. Walter. A policy language for distributed usage control. In Joachim Biskup and Javier López, editors, *12th European Symposium On Research In Computer Security (ESORICS)*, volume 4734 of *Lecture Notes in Computer Science*, pages 531–546, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[32] Health insurance portability and accountability act (HIPAA), 1996.

[33] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. Ryoan: A distributed sandbox for untrusted computation on secret data. In *OSDI*, pages 533–549, 2016.

[34] Mohammed Hussain and David Skillicorn. Persona-based identity management: A novel approach to privacy protection. In *Proceedings of the 13th Nordic Workshop on Secure IT Systems*, WWSecIT '08, pages 201–212, 2008.

[35] Sampo Kellomaki and Rob Lockhart. Liberty ID-SIS personal profile service specification. `http://projectliberty.org/liberty/content/download/1028/7146/file/liberty-idsis-pp-v1.1.pdf`.

[36] David P. Kormann and Aviel D. Rubin. Risks of the Passport single sign-on protocol. *Comput. Netw.*, 33(1-6):51–58, June 2000.

[37] Elisavet Kozyri and Fred B. Schneider. RIF: Reactive information flow labels. Technical report, Cornell University, Computing and Information Science. In preparation.

[38] Susan Landau and Tyler Moore. Economic tussles in federated identity management. In *Tenth Workshop on the Economics of Information Security*, WEIS '11, 2011.

[39] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. Inferring fine-grained control flow inside SGX enclaves with branch shadowing. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 557–574, Vancouver, BC, 2017. USENIX Association.

[40] Peng Li and Steve Zdancewic. Downgrading policies and relaxed noninterference. In *ACM SIGPLAN Notices*, volume 40, pages 158–170, 2005.

[41] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, and Sumit Shah. First experiences using XACML for access control in distributed systems. In *Proceedings of the 2003 ACM workshop on XML security*, pages 25–37. ACM, 2003.

[42] Paul Madsen. Liberty ID-FF authentication context specification. `http://projectliberty.org/liberty/content/download/1209/7927/file/draft-liberty-authentication-context-v2.0-01.pdf`.

[43] Frank McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM, 2009.

[44] Microsoft .NET Passport. `http://www.passport.net`.

[45] Marco Casassa Mont, Siani Pearson, and Pete Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In *Proceedings of the 14th IEEE International Workshop on Database and Expert Systems Applications*, pages 377–382, 2003.

[46] Craig Mundie. Privacy pragmatism: Focus on data use, not data collection. *Foreign Aff.*, 93:28, 2014.

[47] Helen Nissenbaum. *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford University Press, 2009.

[48] Helen Nissenbaum. A contextual approach to privacy online. *Daedalus*, 140(4):32–48, 2011.

[49] OAuth 2.0. `http://tools.ietf.org/html/draft-ietf-oauth-v2-31`.

[50] Openid. `http://openid.net/`.

[51] Christian Paquin. U-Prove technology overview v1.1. `http://research.microsoft.com/pubs/166980/U-ProveTechnologyOverviewV1.1.pdf`, February 2011.

[52] Raúl Pardo, Christian Colombo, Gordon J. Pace, and Gerardo Schneider. An automata-based approach to evolving privacy policies for social networks. In *Proceedings of the 16th International Conference on Runtime Verification (RV 2016)*, volume 10012 of *Lecture Notes in Computer Science*, pages 285–301. Springer International Publishing, 2016.

[53] Jaehong Park and Ravi Sandhu. Towards usage control models: Beyond traditional access control. In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*, SACMAT '02, pages 57–64, 2002.

[54] Jaehong Park and Ravi Sandhu. The UCONABC usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, February 2004.

[55] Milan Petkovic, Davide Prandi, and Nicola Zannone. Purpose control: Did you process the data for the intended purpose? *Secure Data Management*, 6933:145–168, 2011.

[56] PMSys. `http://forzasys.com/pmsys.html`.

[57] Alexander Pretschner, Manuel Hilty, and David Basin. Distributed usage control. *Communications of the ACM*, 49(9):39–44, 2006.

[58] PRIME: Privacy and identity management for Europe. `http://www.prime-project.eu`.

[59] Project liberty. `http://projectliberty.org`.

[60] N. Ramanathan, F. Alquaddoomi, H. Falaki, D. George, C. K. Hsieh, J. Jenkins, C. Ketcham, B. Longstaff, J. Ooms, J. Selsky, H. Tangmunarunkit, and D. Estrin. Ohmage: An open mobile system for activity and experience sampling. In *2012 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 203–204, May 2012.

[61] Fred B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(1):30–50, 2000.

[62] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. VC3: Trustworthy data analytics in the cloud using SGX. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 38–54. IEEE, 2015.

[63] Shayak Sen, Saikat Guha, Anupam Datta, Sriram K. Rajamani, Janice Tsai, and Jeannette M. Wing. Bootstrapping privacy compliance in big data systems. In *Proceedings of the 35th IEEE Symposium on Security and Privacy (Oakland)*, 2014.

[64] Shibboleth. `http://shibboleth.internet2.edu/shib-v2.0.html`.

[65] David Skillicorn and Mohammed Hussain. Personas: Beyond identity protection by information control. A Report to the Privacy Commissioner of Canada, March 2009.

[66] Spring. Spring boot framework. `https://projects.spring.io/spring-boot/`, December 2017.

[67] Hongsuda Tangmunarunkit, Cheng-Kang Hsieh, Brent Longstaff, S Nolen, John Jenkins, Cameron Ketcham, Joshua Selsky, Faisal Alquaddoomi, Dony George, Jinha Kang, et al. Ohmage: A general and extensible end-to-end participatory sensing platform. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):38, 2015.

[68] Technical information on Bluewin identity provider. http://projectliberty.org/liberty/content/download/378/2693/file/IdP_Public_TechWhitePaper_Englishv2.3.pdf, June 2004.

[69] Virtualbox. https://www.virtualbox.org/.

[70] Rod Widdowson and Scott Cantor. OASIS identity provider discovery service protocol and profile. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-idp-discovery.pdf, March 2008.

[71] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 640–656. IEEE, 2015.

[72] Fan Zhang. mbedtls-SGX. https://github.com/bl4ck5un/mbedtls-SGX.